(1)

NCS TIB 87-8

NATIONAL COMMUNICATIONS SYSTEM

TECHNICAL INFORMATION BULLETIN 87-8

TRANSFORM CODING AND DIFFERENTIAL PULSE CODE MODULATION FOR GROUP 4 FACSIMILE

AUGUST 1987

REPORT DOCUMENTATION PAGE

1. AGENCY USE ONLY

2. REPORT DATE: August 1987

3. REPORT TYPE AND DATES COVERED: Final

4. TITLE AND SUBTITLE

Transform Coding and Differential Pulse Code Modulation for Group 4 Facsimile

5. FUNDING NUMBERS

C-DCA100-83-C-0047

6. AUTHOR(S)

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Delta Information Systems, Inc.
Horsham Business Center, Bldg. 3
300 Welsh Road
Horsham, PA 19044

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING MONITORING AGENCY NAME(S) AND ADDRESS(ES)

National Communications System
Office of Technology & Standards
Washington, DC 20305-2010

10. SPONSORING MONITORING AGENCY REPORT NUMBER

NCS TIB 87-8

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION AVAILABILITY STATEMENT

Approved for Public Release; Distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

This document comparies Transform Coding with Differential Pulse Code Modulation (DPCM) in order to determine the relative effectiveness of each technique as applied to the compression of gray scale images for Group 4 facsimile. At the present time, the CCITT Recommendations for Group 4 facsimile permits the transmission of black-white imagery only. Consequently, any input page containing gray scale information, such as a photograph, will be severely distorted by basic Group 4 machines. However, there are plans by the CCITT to add a Gray Scale option to the Group 4 facsimile standard for transmitting pictorial data.

Keywords

14. SUBJECT TERMS

Differential Pulse Code Modulation (DPCM), Group 4, Gray Scale, Facsimile, Transform Coding

15. NUMBER OF PAGES: 165

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

NCS TECHNICAL INFORMATION BULLETIN 87-8

TRANSFORM CODING AND DIFFERENTIAL PULSE CODE MODULATION FOR GROUP 4 FACSIMILE

PROJECT OFFICER

Dennis Bodson

DENNIS BODSON
Senior Electronics Engineer
Office of NCS Technology
and Standards

APPROVED FOR PUBLICATION:

Dennis Bodson

DENNIS BODSON
Assistant Manager
Office of NCS Technology
and Standards

FOREWORD

Among the responsibilities assigned to the Office of the Manager, National Communications System, is the management of the Federal Telecommunication Standards Program. Under this program, the NCS, with the assistance of the Federal Telecommunication Standards Committee identifies, develops, and coordinates proposed Federal Standards which either contribute to the interoperability of functionally similar Federal telecommunication systems or to the achievement of a compatible and efficient interface between computer and telecommunication systems. In developing and coordinating these standards, a considerable amount of effort is expended in initiating and pursuing joint standards development efforts with appropriate technical committees of the Electronics Industries Association, the American National Standards Institute, the International Organization for Standardization, and the International Telegraph and Telephone Consultative Committee of the International Telecommunication Union. This Technical Information Bulletin presents an overview of an effort which is contributing to the development of compatible Federal, national, and international standards in the area of facsimile. It has been prepared to inform interested Federal activities of the progress of these efforts. Any comments, inputs or statements of requirements which could assist in the advancement of this work are welcome and should be addressed to:

Office of the Manager
National Communications System
ATTN: NCS-TS
Washington, DC 20305-2010

COMPUTER SIMULATION OF TRANSFORM CODING FOR GROUP 4 FACSIMILE

August, 1987

Final Report

Submitted to:

NATIONAL COMMUNICATIONS SYSTEM

Office of Technology and Standards

Washington, DC 20305

Contracting Agency:

DEFENSE COMMUNICATIONS AGENCY

Contract Number - DCA100-83-C-0047

Modification/Task Number - P00009/2

D E L T A I N F O R M A T I O N S Y S T E M S, I N C.

Horsham Business Center, Bldg. 3

300 Welsh Road

Horsham PA 19044

Table of Contents

1.0 INTRODUCTION

This document summarizes work performed by Delta Information Systems, Inc., for the Office of Technology and Standards of the National Communications System, an organization of the U. S. Government, headed by National Communications System Assistant Manager for the Office of Technology and Standards, Dennis Bodson. Mr. Bodson is responsible for the management of the Federal Telecommunications Standards Program, which develops telecommunications standards, the use of which is mandatory for all Federal agencies. The purpose of this study, performed under Task 2 of Modification Number P00009 of contract number DCA100-83-C-0047, was to compare Transform Coding with Differential Pulse Code Modulation (DPCM) in order to determine the relative effectiveness of each technique as applied to the compression of gray scale images for Group 4 facsimile.

At the present time, the CCITT Recommendations for Group 4 facsimile permit the transmission of black-white imagery only. Consequently, any input page containing gray scale information, such as a photograph, will be severely distorted by basic Group 4 machines. However, there are plans by the CCITT to add a gray scale option to the Group 4 facsimile standard for transmitting pictorial data.

Both Differential Pulse Code Modulation (DPCM) and transform coding techniques have been used with some success to compress pictorial (gray scale) data. Each of these techniques has some attractive characteristics and some limitations. Transform

coding systems achieve superior performance at high compression, and show less sensitivity to picture data statistics compared to DPCM systems. On the other hand, DPCM systems achieve better performance at lower compression and are less complex to implement, as compared to transform coding systems.

This report is comprised of four sections. Section 1.0 provides a brief description of the objectives of the study and contains a synopsis that outlines the results obtained and conclusions made. Section 2.0 presents the technical approach employed in the study and includes a discussion of gray scale compression techniques, detailed descriptions of the transform coding algorithms simulated, and a discussion of the test image selection process. The results of the simulation study are presented in Section 3.0, and the conclusions and recommendations made based on these results are contained in Section 4.0.

1.1 Synopsis

Transform coding algorithms generally consist of two basic steps, the transformation step and the sub-block coding step. In the transformation step, the image is first divided into sub-blocks of (NxN) pixels each (in this study N=16); each sub-block is then transformed from a set of gray level values into a set of coefficients by applying to it a linear transformation such as the Fourier transform. In this study, it was determined that the type of transform employed had less of an impact on image

compression than the sub-block coding technique employed. An analysis of the available transforms, based on complexity of implementation and overall performance, was performed; the Discrete Cosine transform (DCT) was selected as the transform to be employed in simulating four transform coding algorithms, each of which employs a different sub-block coding technique.

Four sub-block coding techniques were then selected from among the many available algorithms of this type. The conditional zonal coding technique compresses an image by discarding all but a pre-determined number of coefficients within each sub-block (i.e. those in a specified "zone" of the sub-block) and then further quantizing the retained coefficients. The adaptive zonal coding technique is a variation of the conditional zonal coding technique; it adds the element of image dependency in that it determines the number of coefficients retained in each sub-block based on the local image statistics.

The basic Chen-Smith coding technique is more complex than the two zonal coding techniques in that it requires two passes over an image in order to compress it. In the first pass, statistical information is gathered in order to characterize the image; in the second pass, these statistics are employed in order to assign code bits to the coefficients in each sub-block. The image dependent Chen-Smith coding technique is a variation of the basic Chen-Smith coding technique that adds image dependency to the compression process. The effect of this image dependency is that more coding bits are assigned to the more active regions of

the image and fewer coding bits are assigned to the less active regions of the image. At a given target compression, the image dependency improves the image quality with images containing a significant amount of activity and improves the achieved compression with less active images.

Two DPCM compression algorithms were simulated in a previous study performed by Delta Information Systems; the first, conditional DPCM, employs a three-neighbor gray level value predictor, a non-linear three-bit quantizer, Huffman entropy coding, and an optional staggered horizontal subsampler and corresponding interpolator; the second, adaptive DPCM, employs a three neighbor gray level value predictor, an extended non-linear five-bit quantizer, adaptive arithmetic coding, and optional horizontal and vertical spatial filters.

The image dependent Chen-Smith coding algorithm produced the best overall image quality of the four transform coding algorithms, followed by the basic Chen-Smith, adaptive zonal, and conditional zonal coding algorithms. The DPCM algorithms produced image quality comparable to the transform coding algorithms at bit rates above 1 bit/pixel, and performed slightly better than the transform coding algorithms at bit rates as low as 0.63 bits/pixel. However, the DPCM algorithms could not achieve compression below 0.63 bits/pixel; the transform coding algorithms offer the advantage of selectable compression, and thus can reach much lower bit rates (0.10 bits/pixel in this study).

The DPCM algorithms are much less complex than the transform coding algorithms in terms of implementation, and produce very good image quality at relatively low bit rates. DPCM algorithms should be considered in applications where ease of implementation, moderate compression, and good image quality are required. The transform coding algorithms are much more flexible than the DPCM algorithms parametrically; they can be modified easily to suit changing performance requirements. Transform coding algorithms should be considered in applications where the tradeoff between image quality and compression is variable, and ease of implementation is not critical.

2.0 TECHNICAL APPROACH

2.1 Compression Techniques

Figure 2.1 illustrates the wide range of gray scale coding techniques which could be employed in implementing a gray scale option for Group 4 facsimile. Two of these techniques, differential pulse code modulation (DPCM) and transform coding, were compared in this study. Simulations of several DPCM algorithms were performed by Delta Information Systems in a previous study (Ref. 4); the results of those simulations were used in this study for comparison purposes. The simulation effort in this study was therefore centered on the transform coding algorithms to be discussed shortly.

Transform coding algorithms, generally speaking, operate as two step processes. The first step involves performing linear transformations on the original signal (separated into sub-blocks of N x N pixels each), in which signal space is mapped into transform space. In the second step, the transformed signal is compressed by encoding each sub-block through quantization. The reconstruction operation involves performing an inverse transformation of each decoded transformed sub-block. The function of the transformation operation is to make the transformed samples more independent than the original samples, so that the subsequent operation of quantization may be done more efficiently.

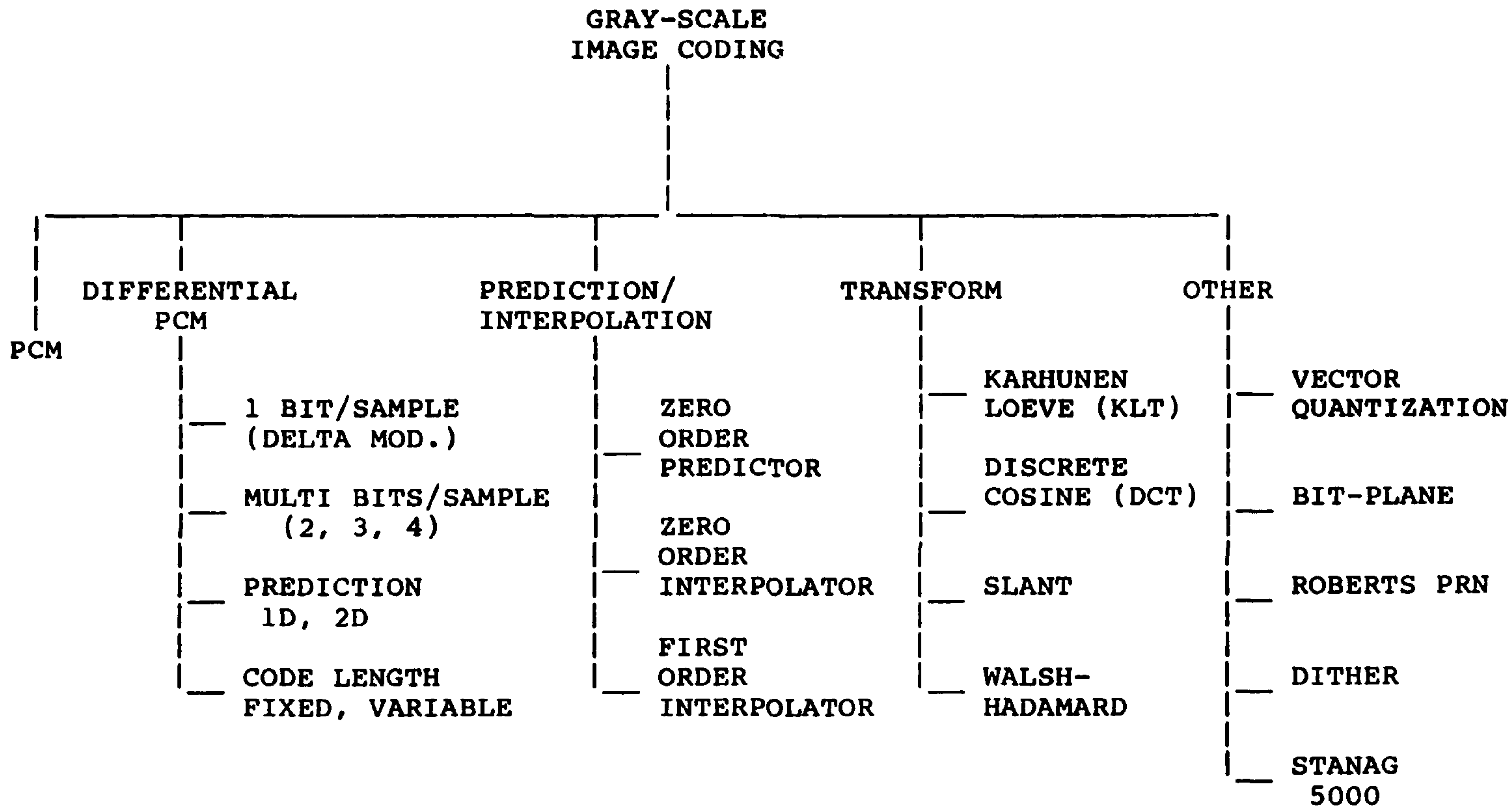

FIGURE 2.1 GRAY SCALE CODING TECHNIQUES

The transformation operation itself does not provide compression; rather, it is a re-mapping of the signal into another domain in which compression can be achieved more easily. It is for this reason that the specific type of transform used will have less of an impact on image quality and compression than the efficient selection of coefficients to be retained and the number of bits allocated to them (i.e., quantization). Therefore, somewhat greater emphasis was put on evaluating sub-block coding techniques, which have more of an impact on image quality and data compression, than the transformation techniques themselves.

2.1.1 Transformation Techniques

Transforms that have proven useful include the Karhunen-Loeve, Discrete Fourier, Discrete Cosine, and Walsh-Hadamard transforms. These transformation techniques were investigated in this study in order to select one particular transform technique to be used in the simulation effort. The selection was based on the overall performance and relative complexity of each candidate technique.

Karhunen-Loeve Transform

The Karhunen-Loeve transform (KLT) is considered to be an optimum transformation, and for this reason many other

transformations have been compared to it in terms of performance. However, the KLT has certain characteristics that make it less than ideal for image processing. These include the necessity to estimate the covariance matrix before processing in both row and column processing operations. Also, the actual eigenvector determination must be carried out to generate the basis matrix. These drawbacks would not be significant if the efficiency of the KLT was much greater those that of other transforms. However, for data having high inter-element correlation, the performance of other transforms (such as the Discrete Cosine transform) is virtually indistinguishable from that of the KLT, and thus does not warrant its added complexity. Therefore, the KLT was not chosen for investigation in this study.

Discrete Fourier Transform

The Discrete Fourier transform is one of the few complex transforms used in data coding schemes. There are disadvantages in using a complex transform for data coding, the most obvious of which is the storage and manipulation of complex numbers. Again, as in the case of the KLT, this complexity issue would not be a factor if the performance of the DFT was significantly greater than that of other transforms. However, other transforms which are less complex perform better than the DFT. For this reason, the DFT was not investigated in this study.

Discrete Cosine Transform

The discrete cosine transform (DCT) is one of an extensive family of sinusoidal transforms. In their discrete form, the basis vectors consist of sampled values of sinusoidal or cosinusoidal functions that, unlike those of the DFT, are real number quantities. The DCT has been singled out for special attention by workers in the image processing field, principally because, for conventional image data having reasonably high inter-element correlation, the DCT's performance is virtually indistinguishable from that of other transforms which are much more complex to implement. Because of its excellent performance and comparatively simple implementation, the Discrete Cosine Transform was chosen for evaluation in this study.

Walsh-Hadamard Transform

The three transforms mentioned previously have basic functions which are either cosinusoidal, i.e. the Fourier and Discrete Cosine, or are a good approximation of a sinusoidal function, such as the Karhunen-Loeve Transform. The Walsh-Hadamard Transform is an approximation of a rectangular orthonormal function. The actual transform consists of a matrix of +1 and -1 values, which eliminates multiplications from the transform process. The elimination of multiplications is a significant property, since the aforementioned transforms require

real or complex multiplications. However, the Walsh-Hadamard transform does not provide the excellent performance that the Discrete Cosine Transform provides. Therefore, the Walsh-Hadamard transform was not chosen for evaluation in this study.

2.1.2 Sub-Block Coding Techniques

Perhaps more important than choosing a specific transform method for image processing is choosing a method for coding the matrix coefficients after transformation. Of the many coefficient coding schemes discussed in the literature, four were selected for evaluation in this study. The first is a simple, non-adaptive, conditional zonal coding technique which uses a fixed number of bits to encode an image. The second is an adaptive, one-pass, image dependent zonal method which uses as many bits as necessary to code a particular image.

The third and fourth sub-block coding algorithms selected are variations of an algorithm developed by Chen and Smith (Ref. 1), which are significantly more complex than the two zonal methods. The basic Chen-Smith technique is adaptive, as is the adaptive zonal technique; however, two passes over the image are necessary. In the first pass, statistics are gathered and bit maps are produced. In the second pass the image is actually coded for transmission. The second variation of the basic Chen-Smith algorithm adds image dependency to the coding scheme. All of the coding techniques evaluated in this report use a (16x16)

Discrete Cosine transform. The following paragraphs describe the coding techniques in more detail.

Conditional Zonal Technique

In conditional zonal coding, all coefficients in a sub-block that are outside a specified zone (usually the upper left hand corner of the sub-block) are discarded prior to the quantization step. The number of coefficients retained per sub-block is selected based on the compression desired; this number remains constant for all sub-blocks in the image. After the significant coefficients have been extracted from the sub-block, they are normalized and quantized to a fixed number of bits through various arithmetic operations based on general image statistics. At the receiver, arithmetic operations to reverse the normalization process are performed to produce reconstructed transform coefficients (with quantization error) in the specified zone of the sub-block; all of the coefficients discarded in the encoding process are set to zero, and the reconstructed sub-block is ready to be inversely transformed.

This technique is extremely simple and requires a minimal amount of overhead as compared to the Chen-Smith techniques to be discussed shortly. The conditional zonal technique can be thought of as being on the simple end of the complexity spectrum, while the Chen-Smith techniques are on the complex end. This technique was investigated in this study for that reason; that

is, to compare a simple and a complex coefficient coding technique in terms of compression and image quality.

Adaptive Zonal Technique

The adaptive zonal coding technique is a combination of threshold coding and conditional zonal coding. In straight threshold coding, a specific energy amplitude is selected, and only those transform coefficients in a sub-block that are above this threshold value are retained; the other coefficients are discarded. A major disadvantage to threshold coding is the overhead required to store information regarding the location within the sub-block of the coefficients which are retained. Zonal coding, as described above, quantizes only those coefficients in a specified area, or zone; because the positions of the retained coefficients are known, the information concerning their locations need not be stored.

The adaptive zonal technique is a hybrid scheme providing the benefits of both zonal and threshold coding. Coefficients in a specified zone are compared to a selected threshold value in an ordered pattern until a coefficient value below the threshold is encountered. When a coefficient below the selected threshold is encountered, the remaining coefficients in the specified zone are discarded, and the retained coefficients are normalized and quantized as in the conditional zonal coding technique. The only additional overhead required by the adaptive zonal coding

technique is to store the number of coefficients retained in each sub-block. The adaptive zonal coding technique achieves superior performance in terms of compression over the conditional zonal coding technique by eliminating some of the trivial coefficients that would be unnecessarily encoded by the conditional technique.

Basic Chen-Smith Technique

The basic Chen-Smith coding technique (Ref. 1) is very popular for coding both monochrome and color images. This technique uses Max's method of optimum quantizer design (Ref. 3), assuming Gaussian DC and AC coefficient probability density functions. Transform sub-blocks of the original image are assigned to one of four classes on the basis of sub-block AC energy. The variance of each coefficient is calculated and used in a bit allocation technique in order to determine a bit assignment map for each class. The transform coefficients are normalized by their corresponding variances to achieve unit variance prior to quantization. The basic Chen-Smith approach is designed to achieve a given compression no matter what image is to be compressed. This means that, for a given compression, the image qualitv of more complex images is poorer than that of less complex images.

Image Dependent Chen-Smith Technique

In addition to the standard Chen-Smith technique described above, a variation of this technique was evaluated in this study. This variation adds image dependency to the technique by analyzing all of the AC energies of sub-blocks in the image in order to allocate more bits to busy sub-blocks. More bits are allocated per sub-block to images with a high amount of activity, and fewer bits per sub-block are allocated to images with a low amount of activity in order to achieve higher compression for images with low activity, and better image quality for images with high activity.

2.2 Algorithm Descriptions

The algorithms for the Discrete Cosine Transform (DCT) and the four sub-block coding techniques are described in this section. The DCT algorithm which was used in each of the four sub-block coding techniques is described first, followed by the descriptions of the four sub-block coding techniques. The software documentation for all simulation software is presented in Appendix A.

2.2.1 Discrete Cosine Transform

The implementation of the Discrete Cosine Transform algorithm requires the division an image into ? series of (NxN) sub-blocks of pixels. Each sub-block is transformed by a two dimensional (NxN) Discrete Cosine Transform process as follows:

$$[T] = [C]\cdot[D]\cdot[C]^T \quad ,$$

where [T] is the transformed sub-block, [C] is the DCT basis matrix, and [D] is the input data sub-block ($[C]^T$ is the transpose of the DCT basis matrix). The DCT basis matrix coefficients were determined from the following relation:

$$C_{i,j} = C_o\cdot\sqrt{(2/N)}\cdot(\cos(i\cdot(j + 0.5)\cdot(\pi/N))) \quad ,$$

where $C_o = 1/\sqrt{2}$ for $i = 0$, $C_o = 1$ otherwise, and $i=j=0$ to $N-1$. This transformation converts each (NxN) sub-block of pixels into an (NxN) matrix of transform coefficients, which consists of one DC coefficient and (NxN - 1) AC coefficients. The sum of the squares of all of the AC coefficients in a given transform matrix is known as the AC energy of that transform matrix, and will be referred to as such throughout this report.

The size of the (NxN) transform chosen for use in the simulations was (16x16). The (16x16) transform size was chosen primarily because it has been used frequently in past applications in the image processing field. It is also a compromise between an (8x8) transform, which would increase overhead due to the greater number of sub-blocks in an image, and a (32x32) transform, which would increase the complexity of the

system. This (16x16) Discrete Cosine Transform was used in the four coding techniques discussed below.

2.2.2 Conditional Zonal Coding

The conditional zonal coding technique encodes transform coefficients of a particular zone of each image sub-block. The size of the zone used in the algorithm is determined by an input parameter that designates the desired number of coefficients to be retained for quantization. The number of coefficients retained in each sub-block remains constant throughout the encoding of the image, which makes this technique non-adaptive.

When simulations were performed on training images, statistics were gathered on transform coefficients over the entire set of training images. These statistics included the variances of the coefficients, which were employed in order to determine the processing order of the coefficients within the selected zone of the sub-block, and the minimum-maximum values of the coefficients, which were used to normalize and quantize the coefficients for compression purposes. The variances were computed assuming an AC coefficient mean of zero.

The coefficient processing order was determined based on decreasing coefficient variances. This order is much like the classical zig-zag technique (Figure 2.2) with minor variations (Figure 2.3). The reason for the change in order from the zig-zag ordering was that the zig-zag ordering did not exactly match

1	3	4	10	11	21	22	36	37	55	56
2	5	9	12	20	23	35	38	54	57	
6	8	13	19	24	34	39	53	58		
7	14	18	25	33	40	52	59			
15	17	26	32	41	51	60				
16	27	31	42	50	61					
28	30	43	49	62						
29	44	48	63							
45	47	64	70							
46	65	69								
66	68									
67										

Figure 2.2 - Zigzag Order Technique

1	3	6	10	16	25	35	45	56
2	5	9	14	19	24	34	44	60
4	8	12	20	26	32	43	53	62
7	13	17	28	33	41	52	64	
11	18	21	30	39	49	63		
15	23	29	36	48	59	70		
22	31	38	47	57	68			
27	37	42	55	66				
40	46	50	61					
51	58	67						
54								
65								
69								

Figure 2.3 - Ordering Matrix Employed in Zonal Coding

16	13	13	12	10	8	8	5	4
13	13	10	9	9	7	6	5	4
13	10	9	9	8	7	4	5	3
13	10	9	8	8	7	6	4	
12	10	9	7	7	7	6		
10	8	7	8	7	5	3		
10	8	6	7	6	5			
10	7	6	5	4				
5	6	5	3					
5	5	5						
7								
3								
2								

Figure 2.4 - Dividing Factors Employed in Zonal Coding

the order of variances of the DCT. Therefore, in order to optimize the quantization of coefficients, this variation of the zig-zag technique was implemented.

When a number of coefficients is specified for retention in the zonal technique, the zone is determined by starting with the DC coefficient and then proceeding in the order of decreasing variance until the specified number of coefficients is reached. For simulation purposes, the maximum number of coefficients which can be kept in this ordering technique is 70 (It was experimentally determined that all coefficients beyond the 70^{th} were relatively insignificant.). All coefficients which are not quantized are assumed to be zero at the decoder for inverse transform purposes.

The quantization technique used is a uniform 8-bit quantizer. When statistics were gathered on the training images, a minimum-maximum matrix of coefficients was produced showing minimum and maximum coefficient values. Once the minimum and maximum values were known, dividing factors were assigned to each coefficient position (Figure 2.4). When a division is performed for quantization on the coefficients, the results are placed in 8-bit values for transmission (7 bits for data and 1 sign bit). The coefficient reconstruction is performed by multiplying the 8-bit quantized value by the dividing factor for that specific coefficient position.

2.2.3 Adaptive Zonal Coding

The adaptive zonal coding technique employs the same coefficient ordering and coefficient quantization methods used in the zonal technique. Adaptivity is achieved by proceeding in the order shown in Figure 2.3 until a coefficient is encountered which is less than a user-specified AC energy threshold. The ordering system used (by order of variance) is the actual decreasing order of the coefficients in most cases; however, depending on the image data, the actual order may vary from the preset ordering sequence. For this reason, a look-ahead method was devised in order to prevent reaching the threshold (which would terminate the encoding of that sub-block) prematurely if subsequent coefficients were significantly greater in magnitude than the current coefficient being evaluated.

When the AC coefficient threshold is reached, the next two coefficients in the specified order are examined. If both of these coefficients are 50 times greater than the AC coefficient threshold, the processing of the sub-block continues. This type of look-ahead processing was implemented in order to decrease the probability of terminating sub-block encoding before significant transform coefficients are encountered.

2.2.4 Chen-Smith Coding

The Chen-Smith coding technique is a two-pass image coding technique. In the first pass, transform matrix statistics are gathered over the entire image. The statistics-gathering process involves the storage of the AC energies of all sub-blocks in the image, and the variances (the sum of the squares of the coefficients in each position of the transform matrix over the entire image) of the transform coefficients. Once the statistics are gathered, a map of the image is produced (Figure 2.5) using four sub-block classification levels. The map is produced using the AC energies of the sub-blocks, assigning high classification levels (4 or 3) to sub-blocks with high AC energies (i.e. high activity sub-blocks), and low classification levels (1 or 2) to sub-blocks with low AC energies (low activity sub-blocks). The map has (M / 4) entries of each classification level, where M is equal to the number of sub-blocks in the image.

After the classification map is produced, bit allocation maps (Figure 2.6) are generated for each class. The bit allocation maps are produced using a bit allocation function which generates each bit map based upon a specified average amount of bits/coefficient to be used for quantization; a higher number of average bits are allocated to the higher classification levels and a lower number of bits to the lower levels. Since there are an equal number of sub-blocks of each class and each bit allocation map has a fixed number of average bits, the

1	2	2	3	4	4	4	3	2	2	2	3	4	2	1	1	1	2	4	4
3	2	3	4	4	4	4	3	1	3	2	3	2	1	2	2	2	4	4	4
4	2	4	4	4	4	4	3	3	3	3	3	2	1	1	2	3	4	1	1
4	1	4	4	3	3	4	3	3	3	3	2	2	1	2	3	3	1	3	3
4	2	3	3	3	3	4	3	3	3	1	2	1	2	2	1	3	4	3	3
4	2	2	2	4	4	4	3	4	1	2	1	1	1	3	3	4	3	3	3
4	3	3	3	4	4	2	3	2	2	2	1	2	1	2	3	3	3	3	3
4	4	2	4	3	4	3	2	1	2	1	3	2	1	3	4	3	3	3	3
4	4	2	3	4	3	4	1	2	1	2	3	1	1	1	1	3	3	3	3
4	4	2	3	4	3	1	2	1	2	3	1	3	2	2	1	3	2	3	2
4	4	3	2	4	1	2	1	2	3	1	3	1	3	1	3	3	3	1	2
3	4	4	2	1	2	1	2	2	1	3	2	3	3	3	1	1	2	2	2
4	4	4	1	1	2	1	2	2	3	2	3	1	3	2	1	2	1	3	3
4	4	4	2	1	2	2	2	1	2	1	2	3	1	1	2	1	1	2	3
4	4	3	2	3	2	3	1	2	1	1	2	2	2	2	1	1	1	1	4
2	4	2	3	2	2	1	2	3	2	1	2	2	2	1	1	1	1	1	1
3	1	3	2	1	1	2	3	2	3	3	1	3	1	1	1	2	2	1	1
1	3	2	2	1	2	2	1	1	3	2	2	1	1	2	1	2	2	1	3
2	1	2	4	1	4	3	1	3	2	1	1	1	2	3	1	1	1	1	2
2	2	4	4	4	3	3	3	4	1	2	2	1	2	2	1	1	1	1	2

Figure 2.5 - Classification map of a (20 x 20) sub-block image

Note: Each value represents a (16 x 16) pixel sub-block. A 1 specifies a low activity sub-block; a 4 specifies a high activity sub-block.

8765433221110000	8654432221110000	8543322111100000	8332110000000000
7665433221110000	6554432221110000	5544332211000000	3322111100000000
6554433221110000	5544433221110000	4433332211100000	2221111100000000
5544433322111000	4443333222110000	3333322211100000	1111111000000000
4443333222111000	3333333222111000	3332222111100000	1111100000000000
3333322222111100	3333222222111100	2222221111100000	1110000000000000
3333322221111100	2222222221111000	2222211111100000	1000000000000000
2223222111111000	2223321111111100	2111211110000000	0000000000000000
2222322111111100	2223432111110000	1112221110000000	0000000000000000
2222221111111000	1212321111101000	1111111100000000	0000000000000000
1111111111100000	1111111111000000	1111111000000000	0000000000000000
1111111111100000	1111111111000000	1110111100000000	0000000000000000
1111111111100000	1111111111000000	1000001000000000	0000000000000000
1111111111000000	1111111111000000	0000001100000000	0000000000000000
1111111111000000	1111111111000000	0000000100000000	0000000000000000
1111111111000000	1111111122100000	1000000011000000	0000000000000000
Class 4	Class 3	Class 2	Class 1

Figure 2.6 - Bit Allocation Maps of the Chen-Smith Algorithm

8765555500000000	8765555000000000	8655550000000000	8555550000000000
7765555000000000	7765550000000000	6655550000000000	5555500000000000
6555555000000000	5555550000000000	5555500000000000	5555500000000000
6655550000000000	5555500000000000	5555000000000000	5555000000000000
6655550000000000	5555500000000000	5555000000000000	5550000000000000
6555500000000000	5555500000000000	5550000000000000	5550000000000000
5555500000000000	5555000000000000	5550000000000000	5500000000000000
5555000000000000	5555000000000000	5500000000000000	5500000000000000
5555000000000000	5550000000000000	5500000000000000	5500000000000000
5550000000000000	5550000000000000	5500000000000000	5000000000000000
5500000000000000	5500000000000000	5000000000000000	5000000000000000
5500000000000000	5500000000000000	5000000000000000	5000000000000000
5500000000000000	5500000000000000	5000000000000000	0000000000000000
5500000000000000	5500000000000000	5000000000000000	0000000000000000
5500000000000000	5500000000000000	0000000000000000	0000000000000000
5500000000000000	5500000000000000	0000000000000000	0000000000000000
Class 4	Class 3	Class 2	Class 1

Figure 2.7 - Variation of the Bit Allocation Maps Employed in the Chen-Smith Simulation

compression to be achieved using this technique can be a preset run-time parameter. For example, if a 1 bit per pixel compression ratio was desired, the number of average bits for classes 1, 2, 3, and 4 would be .67, .83, 1.17, and 1.33, respectively. In the second pass, the sub-block classification and bit allocation maps are used to encode the image for transmission.

The quantization method used in the Chen-Smith technique is the classical Lloyd-Max quantization technique (Refs. 2,3). This technique is a non-uniform quantization scheme which uses a probability density function (pdf) specific to the distribution of the data to be quantized. In the basic Chen-Smith coding technique, the distribution of transform coefficients is assumed to be Gaussian. Therefore, a Gaussian pdf was used for quantization in the simulations.

A variation of the basic Chen-Smith algorithm involving the generation of bit allocation maps was implemented for the following reason. After preliminary simulations were performed, statistics demonstrated that the quantization of bits/pixel values below 5 in the bit allocation maps would have no positive effect on image quality, and in some cases would degrade image quality. A method was devised which would achieve the same number of bits for the bit allocation map, but would not assign bits/pixel values of less than 5 to any coefficient position. An example of this method is shown in Figure 2.7, which illustrates

the same bit allocation maps shown in Figure 2.6 with the variation implemented.

2.2.5 Image Dependent Chen-Smith Coding

The image dependent Chen-Smith technique is implemented in the same way as the basic Chen-Smith technique, with one variation. The basic Chen-Smith technique is an image independent technique; that is, a preset number of bits is used to encode an image, with an equal number of sub-blocks assigned to each class in the classification map, independent of image characteristics. The image dependent approach is implemented at the time that the image classification map is produced. The AC energies are examined, and, depending on their comparative values, an appropriate number of sub-blocks are assigned each class.

For example, if an active image is processed, the majority of class assignments would be 3's and 4's; if an inactive image is processed, the majority of class assignments would be 1's and 2's. This variation of the Chen-Smith technique is dependent on image characteristics for class assignments (and, thus, the number of total bits for image encoding) and does not necessarily encode a fixed number of bits independent of image characteristics.

2.3 Selection of Test Documents

The test documents employed in the computer simulation were selected based on several factors, including image quality, availability, and feature content. As specified in the statement of work, three gray scale images were chosen. These images are the same three test documents employed in a gray scale study previously performed by Delta Information Systems for the NCS (Ref. 4), in which Differential Pulse Code Modulation (DPCM) and Bit Plane Coding (BPC) were evaluated.

Beyond the advantages these images provide in terms of image quality and availability, each image was selected because it contained several distinctive features that would aid in the subjective evaluation of the output images. The IEEE face image was selected because it contains large areas of relatively smooth tonal range, where artifacts resulting from compression usually manifest themselves. The aerial photo image was chosen because it contains low contrast, high detail regions suitable for visual evaluation of the output images. The crowd scene image contains well-defined structures, such as facial characteristics, which facilitate visual determination of the quality of reproduction.

3.0 RESULTS

3.1 Compression Statistics

The results achieved in the simulations performed to determine the effects of the parametric variations of each of the four transform coding algorithms are summarized in Tables 3.1 through 3.3. Table 3.1, which contains the results of the simulations performed using the IEEE test face image, includes the results of 18 simulation runs, whereas Tables 3.2 and 3.3 include the results for 12 simulation runs each. The IEEE face image was selected to illustrate the visual effects of the compression algorithms; thus, additional simulations were performed with this test image in order to more fully evaluate the effects of the compression techniques on output image quality.

For each simulation run, four statistical measures of performance of the employed algorithm are presented. The first three, the number of compressed bits (the number of bits output by the quantization process), the compression ratio (the number of compressed bits as a function of the number of bits in the input image), and the compressed number of bits per pixel (the effective number of bits per pixel required to transmit the image), provide a measure with which the algorithms can be compared in terms of compression. The fourth measure, the root-mean-square (RMS) error (a weighted-average difference between

TABLE 3.1 - COMPRESSION RESULTS ON THE IEEE FACE

IMAGE	COMPRESSION TECHNIQUE	PIXELS PER LINE	LINES PER IMAGE	ADJUSTABLE PARAMETERS	COMPRESSED BITS	COMPRESSION RATIO	COMPRESSED BITS/PIXEL	RMS ERROR
IEEE FACE	Conditional Zonal Coding	1024	1408	#K 5	225280	51.20	0.16	9.01
				#K 17	765952	15.06	0.53	5.10
				#K 33	1486848	7.76	1.03	3.37
				#K 70	3153920	3.66	2.19	2.06
	Adaptive Zonal Coding	1024	1408	CO 3.00	325032	35.49	0.23	5.52
				CO 1.50	465648	24.77	0.32	4.42
				CO 0.50	849568	13.58	0.59	3.19
				CO 0.04	2345776	4.92	1.63	2.12
	Basic Chen-Smith	1024	1408	BM 0.08	148915	77.46	0.10	8.86
				BM 0.15	299217	38.54	0.21	6.23
				BM 0.50	777410	14.84	0.54	4.06
				BM 1.00	1475685	7.82	1.02	2.87
				BM 2.00	3332747	3.46	2.31	1.83
	Chen-Smith Image Dependent	1024	1408	BM 0.15	202774	56.88	0.14	7.14
				BM 0.30	320570	35.98	0.22	5.78
				BM 1.00	997840	11.56	0.69	3.32
				BM 1.40	1412863	8.16	0.98	2.72
				BM 2.50	2585861	4.46	1.79	1.96

TABLE 3.2 - COMPRESSION RESULTS ON THE CROWD SCENE

IMAGE	COMPRESSION TECHNIQUE	PIXELS PER LINE	LINES PER IMAGE	ADJUSTABLE PARAMETERS	COMPRESSED BITS	COMPRESSION RATIO	COMPRESSED BITS/PIXEL	RMS ERROR
CROWD SCENE	Conditional Zonal Coding	1024	1408	#K 16	720896	16.00	0.50	3.67
				#K 32	1441792	8.00	1.00	2.56
				#K 65	2928640	3.94	2.03	1.89
	Adaptive Zonal Coding	1024	1408	CD 1.00	737672	15.64	0.51	3.55
				CD 0.25	1462680	7.89	1.01	2.49
				CD 0.05	2626904	4.39	1.82	1.96
	Basic Chen-Smith	1024	1408	BM 0.50	768595	15.01	0.53	2.72
				BM 1.00	1485091	7.77	1.03	1.92
				BM 2.00	2982828	3.87	2.07	1.39
	Chen-Smith Image Dependent	1024	1408	BM 0.70	867764	13.29	0.60	2.54
				BM 1.00	1177471	9.80	0.81	2.17
				BM 2.30	2902867	3.97	2.01	1.45

TABLE 3.3 - COMPRESSION STATISTICS ON THE AERIAL PHOTO

IMAGE	COMPRESSION TECHNIQUE	PIXELS PER LINE	LINES PER IMAGE	ADJUSTABLE PARAMETERS	COMPRESSED BITS	COMPRESSION RATIO	COMPRESSED BITS/PIXEL	RMS ERROR
AERIAL PHOTO	Conditional Zonal Coding	1024	1408	#K 15	675840	17.07	0.47	6.27
				#K 33	1486848	7.76	1.03	3.17
				#K 70	3153920	3.66	2.19	2.05
	Adaptive Zonal Coding	1024	1408	CD 1.00	1073856	10.74	0.74	5.16
				CD 0.45	1557064	7.41	1.08	3.73
				CD 0.20	2138528	5.39	1.48	2.80
	Basic Chen-Smith	1024	1408	BM 0.50	747479	15.43	0.51	4.32
				BM 1.00	1552633	7.43	1.08	2.58
				BM 2.00	3334688	3.46	2.31	1.62
	Chen-Smith Image Dependent	1024	1408	BM 0.55	719764	16.03	0.50	4.60
				BM 1.10	1614243	7.14	1.12	2.58
				BM 2.20	2866510	4.02	1.99	1.81

the gray level value of an original input pixel and the corresponding pixel in the decoded output image), provides a basis upon which the algorithms can be compared quantitatively in terms of image quality.

The RMS error is a quantitative measure of the image quality of the output image and is calculated as follows:

$$RMS = \sqrt{\frac{e_1^2 + e_2^2 + \ldots + e_N^2}{N}}$$

where e_i is the 8-bit difference, or error, between the i^{th} pixel in the input image and the corresponding i^{th} pixel in the decoded output image, and N is the total number of pixels in the processed image. The RMS error can also be expressed as a percentage of the dynamic range (2^n, where n = number of bits/input pixel) of the gray scale of the image.

Each transform coding algorithm has an adjustable parameter that can be varied in order to select a target compression; listed below are the abbreviations used in Tables 3.1 through 3.3 to distinguish these parameters:

Abbreviation	Description
#K	Used in the one-pass conditional zonal algorithm to select the number of coefficients to be kept in the quantization zone of each sub-block.
CO	Used in the one-pass adaptive zonal algorithm as a cutoff threshold for the elimination of insignificant

	coefficients prior to quantization.
BM	Both the basic and the image dependent Chen-Smith algorithms assign bits to each class for quantization. BM is used to select the average number of bits per pixel over the four bit map classifications.

The simulations performed to evaluate the conditional zonal coding algorithm were designed so that the effects of the parametric variations were clearly illustrated; the parameter chosen for evaluation in the conditional zonal coding simulations was the number of retained coefficients, or zone size (#K). The effect of the zone size parameter on compression is straightforward; as it is decreased, the number of compressed bits/pixel is decreased. Image content has no effect on the compression achieved by the zonal coding technique; the same number of bits is used to encode each sub-block regardless of the statistics of the sub-block. Simulations in which the zone size was varied were performed in order to determine the parameter's effect on output image quality; the compressions achieved were selected so as to be comparable to the compressions achieved in the DPCM simulations performed in a previous study (Ref. 4).

The simulations performed to evaluate the adaptive zonal coding algorithm were designed so that the effects of the parametric variations were clearly illustrated; the parameter chosen for evaluation in the adaptive zonal coding simulations was the coefficient cutoff threshold (CO). The cutoff thresholds

employed in the simulations were selected in order to achieve compressions comparable to those achieved for the DPCM and conditional zonal coding simulations. While the target compressions for the conditional zonal coding simulations could be precisely selected with the zone size parameter (#K), the target compressions for the adaptive zonal coding simulations were more difficult to select because of the statistical dependency of the technique.

The simulations performed to evaluate the basic Chen-Smith coding algorithm were designed so that the effects of the parametric variations were clearly illustrated; the parameter chosen for evaluation in the basic Chen-Smith coding simulations was the average number of bits/pixel over the four bit map classifications (BM). The average bits/pixel values employed in the basic Chen-Smith simulations were selected so as to produce compression results comparable to those of the other coding techniques evaluated in this study.

The simulations performed to evaluate the image dependent Chen-Smith coding algorithm were designed so that the effects of the parametric variations were clearly illustrated; the parameter chosen for evaluation in the image dependent Chen-Smith coding simulations was the same as that employed in evaluating the basic Chen-Smith coding algorithm, namely the average number of bits/pixel over the four bit map classifications (BM). The average bits/pixel values employed in the image dependent Chen-Smith simulations were selected so as to produce compression

results comparable to those of the other coding techniques evaluated in this study; however, the adaptive nature of this coding technique made it difficult select target compressions as precisely as was possible with the basic Chen-Smith coding technique.

3.2 Output Images

Before the image quality of the transform coding algorithms can be evaluated, an understanding of the type of distortion caused by transform coding is required. The image distortion caused by these algorithms manifests itself in "blocking", in which the edges of the individual sub-blocks become visually apparent. Transform coding algorithms break the image into sub-blocks and process the image one sub-block at a time. Blocking occurs mainly in the busy sections of the images. A large amount of AC energy exists in a busy sub-block, meaning that the transform coefficients of the sub-block contain a large amount of information. Blocking occurs when, through quantization, a significant part of this information is lost, and the reconstructed sub-block in the output image is markedly dissimilar from those sub-blocks surrounding it.

Table 3.4 is a list of the output images presented in Figures 3.2 through 3.18; Figure 3.1 is an illustration of an original input image, the IEEE face. Each image is a photographic reproduction of a windowed portion of the output

TABLE 3.4 - LIST OF OUTPUT IMAGES

FIGURE NUMBER	IMAGE DESCRIPTION
3.1	Windowed portion of Original IEEE Face Image
3.2	Conditional Zonal IEEE Face Image at 0.16 bpp
3.3	Conditional Zonal IEEE Face Image at 0.53 bpp
3.4	Conditional Zonal IEEE Face Image at 1.03 bpp
3.5	Adaptive Zonal IEEE Face Image at 0.32 bpp
3.6	Adaptive Zonal IEEE Face Image at 0.59 bpp
3.7	Adaptive Zonal IEEE Face Image at 1.63 bpp
3.8	Basic Chen-Smith IEEE Face Image at 0.10 bpp
3.9	Basic Chen-Smith IEEE Face Image at 0.54 bpp
3.10	Basic Chen-Smith IEEE Face Image at 1.02 bpp
3.11	Image Dependent Chen-Smith IEEE Face Image at 0.22 bpp
3.12	Image Dependent Chen-Smith IEEE Face Image at 0.69 bpp
3.13	Image Dependent Chen-Smith IEEE Face Image at 0.98 bpp
3.14	Original Circular Test Image
3.15	Conditional Zonal Circular Test Image
3.16	Adaptive Zonal Circular Test Image
3.17	Basic Chen-Smith Circular Test Image
3.18	Image Dependent Chen-Smith Circular Test Image

Figure 3.1 - Windowed portion of Original IEEE Face Image

Figure 3.2 - Conditional Zonal IEEE Face Image at 0.16 bpp

Figure 3.3 - Conditional Zonal IEEE Face Image at 0.53 bpp

Figure 3.4 - Conditional Zonal IEEE Face Image at 1.03 bpp

Figure 3.5 - Adaptive Zonal IEEE Face Image at 0.32 bpp

Figure 3.6 - Adaptive Zonal IEEE Face Image at 0.59 bpp

Figure 3.7 - Adaptive Zonal IEEE Face Image at 1.63 bpp

Figure 3.8 - Basic Chen-Smith IEEE Face Image at 0.10 bpp

Figure 3.9 - Basic Chen-Smith IEEE Face Image at 0.54 bpp

Figure 3.10 - Basic Chen-Smith IEEE Face Image at 1.02 bpp

Figure 3.11 - Image Dependent Chen-Smith IEEE Face Image at 0.22 bpp

Figure 3.12 - Image Dependent Chen-Smith IEEE Face Image at 0.69 bpp

Figure 3.13 - Image Dependent Chen-Smith IEEE Face Image at 0.98 bpp

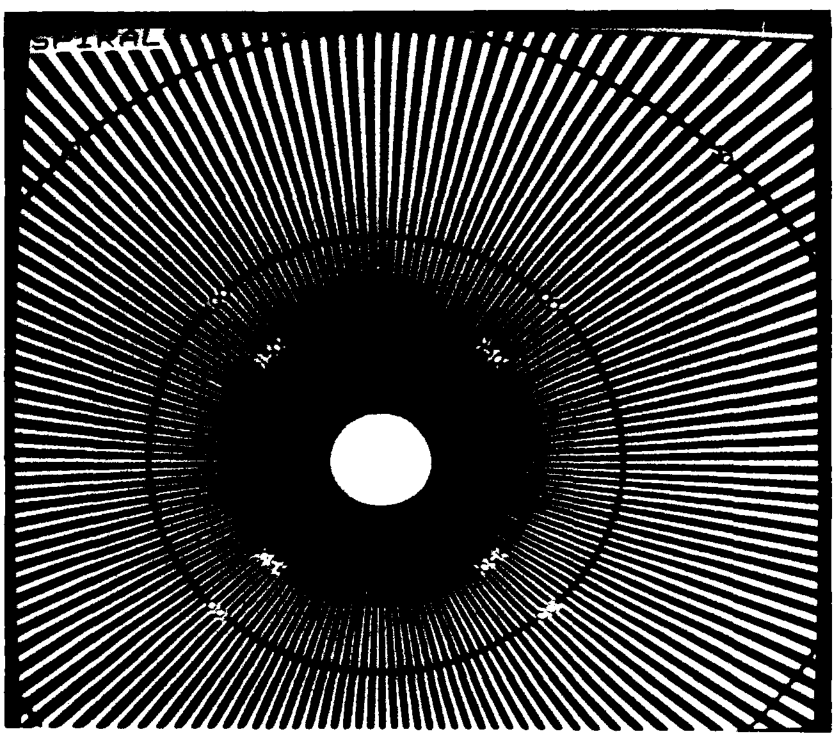

Figure 3.14 - Original Circular Test Image

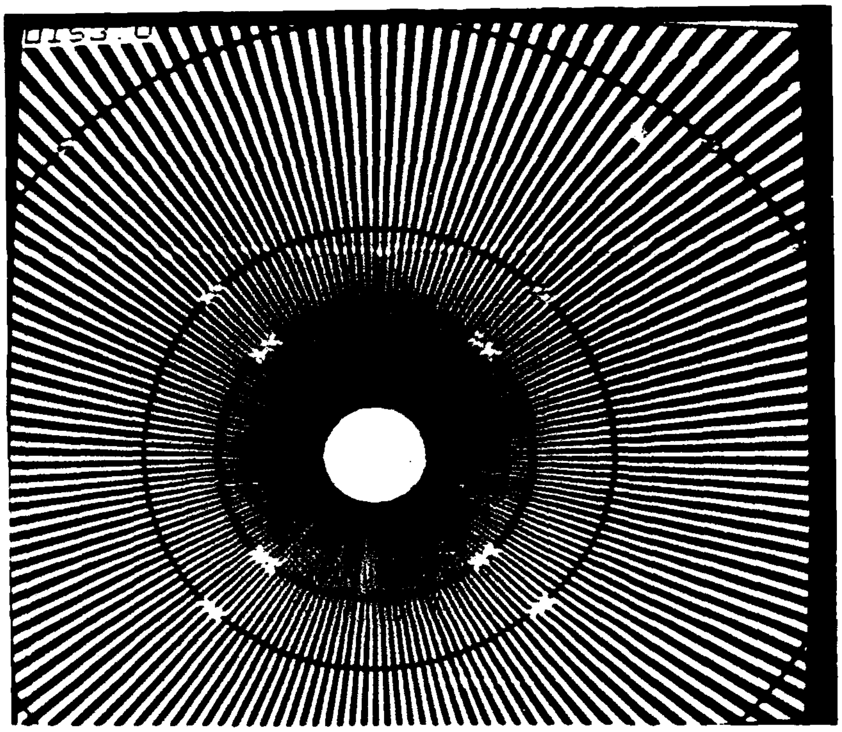

Figure 3.15 - Conditional Zonal Circular Test Image

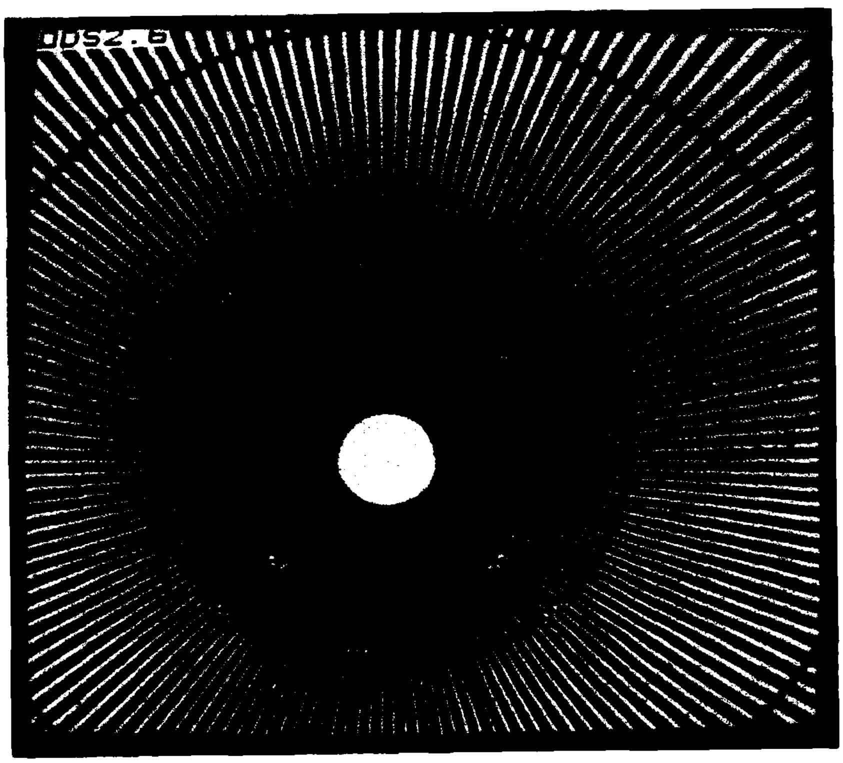

Figure 3.16 - Adaptive Zonal Circular Test Image

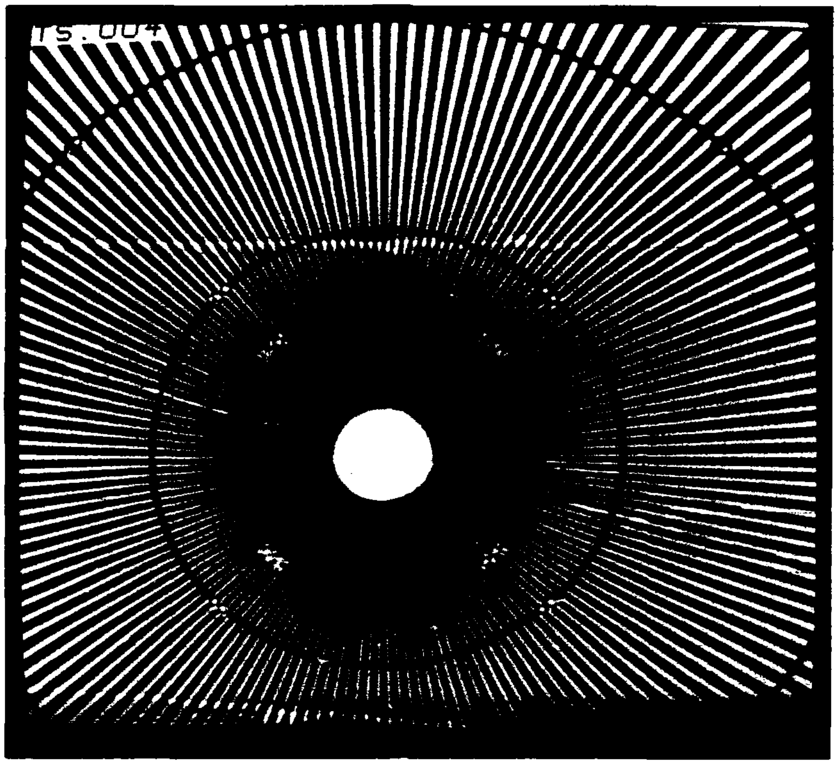

Figure 3.17 - Basic Chen-Smith Circular Test Image

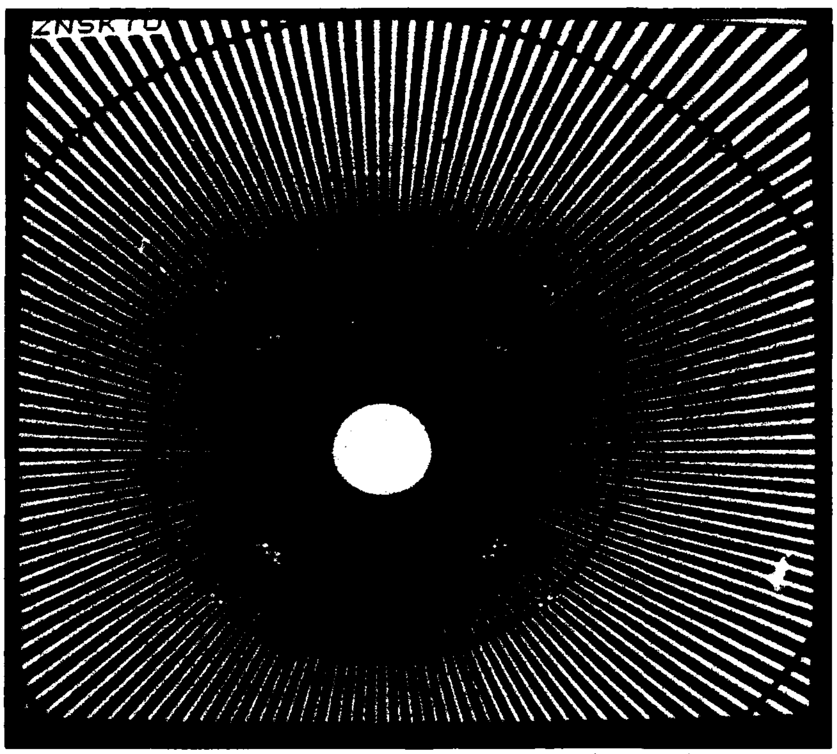

Figure 3.18 - Image Dependent Chen-Smith Circular Test Image

image of one simulation. The full image size was not reproduced photographically because of the limitations of the image storage and display system used to evaluate the output images.

Note that, as in the earlier study (Ref. 4), only the simulations run on the IEEE face image are represented. The effects of the algorithms were similar for all three test images; the IEEE face image was selected as the illustrative example of the output image quality of the algorithms in order to facilitate direct comparisons with the results of the DPCM simulations performed in the earlier study. The evaluation of the image quality of each of the four transform coding algorithms, however, was performed considering the output images from the simulations run on all three test images. In addition, a circular test image, extracted from the IEEE test chart from which the IEEE face image was obtained, was compressed with each transform coding algorithm in order to evaluate the effects of sharp transitions on the image quality produced by the algorithms.

The image quality produced by the transform coding algorithms was generally good above 0.5 bits/pixel and fair at bit rates as low as 0.16 bits/pixel. Quantitatively, the highest RMS error value obtained in the simulations employing the IEEE face image was 9.01, obtained in the conditional zonal simulation run in which 0.16 bits/pixel compression was achieved. This value, measured in gray levels, represents a maximum error of only 3.5 percent of the dynamic range (256 gray levels) of the images. The RMS error, while a good relative measure of the

image quality produced by the algorithms on a particular image, should not be regarded as an absolute measure of image quality. The RMS error is only an average measure of image quality and does not reflect the fact that the algorithms perform well on image regions that are relatively inactive in terms of gray scale activity and not as well on image regions that contain a significant number of gray scale transitions.

The conditional zonal coding algorithm produced images which were very good in terms of image quality for bit rates above 1 bit/pixel. In Figure 3.4, only a minimal amount of blocking can be detected in the high detail regions of the image (e.g. the eyes, the teeth); the overall effect of the blocking is a slight blurring of the image. At lower bit rates, the indiscriminate quantization employed by this coding technique caused significant distortion in the output images. At bit rates on the order of 0.5 bits/pixel, the loss of detail in all areas of the image is evident, and the blocking is much more pronounced (see Figure 3.3). At bits rates below 0.5 bits/pixel, the blocking is severe, and the high detail regions of the image are almost completely degraded (see Figure 3.2).

The adaptive zonal coding algorithm produced images which were excellent in terms of image quality for bit rates above 1 bit/pixel. The image in Figure 3.7 illustrates this level of image quality; no blocking is evident, and only a slight loss of sharpness is detectable in the high detail regions of the image (e.g. the pupils of the eyes). At lower bit rates, blocking

begins to occur in image areas in which moderate gray level transitions are present. Blocking in these areas is caused by excessive quantization; optimization of the look-ahead algorithm would minimize this distortion. At bit rates on the order of 0.5 bits/pixel, some blocking is evident in image regions containing moderate detail (e.g. the nose, the lips); the overall sharpness of the image, however, is only slightly degraded (see Figure 3.6). At bit rates below 0.5 bits/pixel, blocking is evident in areas of moderate to high detail, but the overall image quality is still quite good.

The basic Chen-Smith coding algorithm produced images which were excellent in terms of image quality for bit rates above 1 bit/pixel. The image in Figure 3.10 illustrates this level of image quality; no blocking is evident, and only a slight loss of sharpness is detectable in the high detail regions of the image (e.g. the pupils of the eyes). At bit rates on the order of 0.5 bits/pixel, the image quality is still very good; blocking is only slightly perceptible in the high detail regions of the image, and the overall sharpness of the image is still good (see Figure 3.9). At bit rates below 0.5 bits/pixel, the image quality becomes progressively worse; at 0.1 bits/pixel, the blocking is severe, and the high detail regions of the image are almost completely degraded (see Figure 3.8).

The image dependent Chen-Smith coding algorithm produced images which were excellent in terms of image quality for bit rates above 0.5 bits/pixel. Figure 3.13, in which the image was

compressed to less than 1 bit/pixel, is virtually indistinguishable from the uncompressed image (see Figure 3.1). At bit rates approaching 0.5 bits/pixel, the image quality is still excellent; as Figure 3.12 shows, no blocking is evident, and the overall sharpness of the image is only slightly degraded. At bit rates below 0.5 bits/pixel, the image quality produced by the image dependent Chen-Smith algorithm is still quite good; blocking is evident around the high detail regions of the image, but the overall image quality is still good (see Figure 3.11).

A circular test image, presented in Figure 3.14, was employed to evaluate the performances of the transform coding algorithms; the black-white coloring of the circular test image provided a good test of the quantization functions of the sub-block coding algorithms. In comparing Figures 3.15 through 3.18, it is evident that the image dependent Chen-Smith coding technique produced the best output image; the 100's are still legible in the image dependent Chen-Smith output image, but are quite blurred in the output images of the other compression techniques. This is due to the design of the image dependent Chen-Smith algorithm, which allocates additional coding bits to those image regions which require more information to encode them.

3.3 Algorithm Complexity

In Sections 3.1 and 3.2, the four transform coding algorithms were compared on the basis of compression and image quality. It is also important to compare the coding techniques on the basis of their relative implementation complexities. All four algorithms employ the Discrete Cosine Transform (DCT) in the transformation step; as such, the differences in algorithm complexity occur in the sub-block coding steps of the algorithms.

Of the four techniques, the conditional zonal coding algorithm is the least complex. In the transformation step, the image is divided into sub-blocks, and each sub-block of gray level values is transformed into a matrix of coefficients. In the sub-block coding step, the transform coefficients in a selected zone of the sub-block are normalized and quantized to a selected number of bits, and the remaining coefficients in the sub-block are discarded. Every sub-block within an image is encoded with the same number of bits, and every image is encoded with the same number of bits.

The adaptive zonal coding technique is a hybridization of threshold coding and conditional zonal coding. Threshold coding is a sub-block coding technique in which each coefficient in the sub-block is compared to a specified threshold value in order to determine whether the coefficient is to be kept or discarded. One major drawback to threshold coding is the overhead required to store the locations within the sub-block of each retained

coefficient; adaptive zonal coding eliminates the need for this overhead by performing the threshold comparison in a specific order in a pre-determined zone of the sub-block, thus eliminating the need for the storage of coefficient locations. The only overhead required for adaptive zonal coding is an additional byte of information for each sub-block that indicates the number of coefficients retained in that sub-block; other than that, the encoding proceeds exactly as in the conditional zonal coding technique.

The Chen-Smith coding algorithms are relatively more complex than the zonal coding algorithms; the basic Chen-Smith algorithm processes an image in two passes. The first pass over the image calculates statistics which characterize the image. The statistics are used to determine the number of bits assigned to each coefficient of each sub-block of the image. The AC energies of the sub-blocks are used to produce a sub-block classification map of the image, in which each sub-block is assigned to one of four classes, such that there are an equal number of sub-blocks assigned to each class. A bit allocation map is then produced for each classification, in which the variances of the transform coefficients are used to determine the number of bits to be employed to encode the coefficients. In the second pass, the sub-block classification and bit allocation maps are employed to encode the image.

The image dependent Chen-Smith coding technique is the most complex of the four algorithms simulated. In this variation of

the basic Chen-Smith algorithm, the AC energies of the sub-blocks are used to assign classifications to the sub-blocks based on image content rather than on a pre-specified number of sub-blocks per class. Thus, images containing a high amount of activity are compressed with better output image quality, and images containing a low amount of activity achieve better compression without a significant loss of output image quality.

3.4 DPCM Comparison

Two DPCM compression algorithms were simulated in a study previously performed by Delta Information Systems (Ref. 4). The first, conditional DPCM, employs a three-neighbor gray level value predictor, a non-linear three-bit quantizer, Huffman entropy coding, and an optional staggered horizontal sub-sampler and corresponding interpolator. The second, adaptive DPCM, employs a three neighbor gray level predictor, an extended non-linear five-bit quantizer, adaptive arithmetic coding, and optional horizontal and vertical spatial filters. Quantization in DPCM coding refers to the quantization of the difference between the predicted value of the gray level of a pixel and the actual value.

Table 3.5 summarizes the results of the DPCM simulations which were performed in a previous study (Ref. 4); as can be seen, the same test images employed previously were used in this study in order to make the results directly comparable.

TABLE 3.5 - DPCM COMPRESSION RESULTS

IMAGE	COMPRESSION TECHNIQUE	PIXELS PER LINE	LINES PER IMAGE	ADJUSTABLE PARAMETERS	COMPRESSED BITS	COMPRESSION RATIO	COMPRESSED BITS/PIXEL	RMS ERROR
IEEE FACE	Conditional DPCM	1024	1408	BASE	1880566	6.13	1.30	3.91
				SS	1039426	11.10	0.72	3.86
				SS,HSM	912077	12.65	0.63	4.08
	Adaptive DPCM	1024	1408	BASE	1975109	5.84	1.37	1.20
				HSM	1414586	8.15	0.98	2.00
				HSM,BELL	1444367	7.99	1.00	2.26
CROWD SCENE	Conditional DPCM	1024	1408	BASE	1994002	5.78	1.38	5.32
				SS	1112332	10.37	0.77	4.79
				SS,HSM	968462	11.91	0.67	4.94
	Adaptive DPCM	1024	1408	BASE	2406572	4.79	1.67	1.30
				HSM	1754491	6.57	1.22	2.04
				HSM,BELL	2296846	5.02	1.59	2.14
AERIAL PHOTO	Conditional DPCM	1024	1408	BASE	2144089	5.38	1.49	3.34
				SS	1251099	9.22	0.87	3.33
				SS,HSM	1131161	10.20	0.78	3.75
	Adaptive DPCM	1024	1408	BASE	2943794	3.92	2.04	1.29
				HSM	2286903	5.04	1.59	2.36
				HSM,BELL	2902523	3.97	2.01	2.47

Table 3.6 lists the output images, presented in Figures 3.19 through 3.23, associated with five of the DPCM simulations performed using the IEEE face image. Because the compression achieved in the transform coding simulations was selectable, runs were performed to closely match the compressions achieved by the DPCM algorithms so that direct image quality comparisons could be performed.

The image quality produced by both DPCM algorithms was excellent; the highest RMS error value, obtained in the baseline conditional DPCM simulation run on the crowd scene image, was 5.32. This value, measured in gray levels, represents a maximum error of only 2 percent of the dynamic range (256 gray levels) of the images. The two preprocessing steps employed in the DPCM simulations, horizontal subsampling and horizontal filtering, had the effect of significantly increasing compression while only slightly degrading the output image quality.

The conditional DPCM algorithm without preprocessing produced images which were excellent in terms of image quality; Figure 3.19 illustrates this level of quality. With subsampling (Figure 3.20), the quality of the output images produced by the conditional DPCM algorithm were still quite good, with only a slight blurring effect evident in the high-detail regions of the images (e.g. the hair and teeth regions of the IEEE face image). When both subsampling and horizontal filtering were employed in conjunction with the conditional DPCM algorithm, the image quality illustrated in Figure 3.21 was produced at an encoded bit

TABLE 3.6 - LIST OF DPCM OUTPUT IMAGES

FIGURE NUMBER	IMAGE DESCRIPTION
3.19	Conditional DPCM Encoded IEEE Face Image at 1.30 bpp
3.20	Conditional DPCM Encoded IEEE Face Image with Subsampling at 0.72 bpp
3.21	Conditional DPCM Encoded IEEE Face Image with Filtering and Subsampling at 0.63 bpp
3.22	Adaptive DPCM Encoded IEEE Image at 1.37 bpp
3.23	Adaptive DPCM Encoded IEEE Image with Filtering at 0.98 bpp

Figure 3.19 - Conditional DPCM Encoded IEEE Face Image at 1.30 bpp

Figure 3.20 - Conditional DPCM Encoded IEEE Face Image with Subsampling at 0.72 bpp

Figure 3.21 - Conditional DPCM Encoded IEEE Face Image with Subsampling and Filtering at 0.63 bpp

Figure 3.22 - Adaptive DPCM Encoded IEEE Image at 1.37 bpp

Figure 3.23 - Adaptive DPCM Encoded IEEE Face Image with Filtering at 0.98 bpp

rate of 0.63 bits/pixel. Blurring can be seen in the high detail regions of the hair, and a loss of edge detail is evident in the eye and mouth regions, but the overall quality of this image still quite good.

The adaptive DPCM algorithm without preprocessing produced images which were nearly indistinguishable from the input images; an example of this image quality is presented in Figure 3.22. The adaptive DPCM simulations in which horizontal filtering was employed produced output images which were only slightly less impressive; in observing Figure 3.23, only slight blurring in the hair and eye regions is evident.

The DPCM simulation output images were compared with those produced in the transform coding simulations on the basis of similar compression results. The conditional DPCM encoded images in Figures 3.20 and 3.21 are comparable, in terms of compression, to the conditional zonal coded image in Figure 3.3, the adaptive zonal coded image in Figure 3.6, the basic Chen-Smith coded image in Figure 3.9, and the image dependent Chen-Smith coded image in Figure 3.12. In terms of image quality, the image dependent Chen-Smith coding technique appears to have performed best (Figure 3.12), followed closely by the two conditional DPCM variations (Figures 3.20 and 3.21), the basic Chen-Smith coding technique (Figure 3.9), the adaptive zonal coding technique (Figure 3.6), and the conditional zonal coding technique (Figure 3.3).

At this level of compression (0.5-0.7 bits/pixel), the differences between DPCM and transform coding, in terms of effect on image quality, manifest themselves. The DPCM images appear blurred in the high detail regions of the images, but are free of any compression-induced artifacts. The transform coded images, however, contain artifacts due to blocking (particularly evident in Figure 3.3) in addition to the loss of sharpness in the high detail regions.

At higher compression rates (1-1.3 bits/pixel), the image quality of both the DPCM and the transform coding algorithms was excellent. The conditional DPCM encoded image in Figure 3.19 and the adaptive DPCM encoded images in Figures 2.22 and 2.23 are comparable, in terms of compression, to the conditional zonal coded image in Figure 3.4, the adaptive zonal coded image in Figure 3.7, the basic Chen-Smith coded image in Figure 3.10, and the image dependent Chen-Smith coded image in Figure 3.13. Only the conditional zonal coding technique (Figure 3.4) shows any visually significant image degradation in this compression range; slight blocking is evident in the high detail regions of the image.

Image comparisons could not be performed for bit rates below 0.6 bits/pixel because the lowest bit rate achieved in the DPCM simulations was 0.63 bits/pixel. This is a primary drawback to DPCM compression techniques; the compression achieved is governed by image statistics. Transform coding algorithms employ a target compression parameter that is independent of image statistics,

thus giving transform coding algorithms the flexibility of sacrificing image quality to increase compression and vice versa.

DPCM compression algorithms are, in general, less complex to implement than transform coding algorithms; they require less data storage, are much less demanding computationally, and do not require overhead data such as that associated with many transform coding algorithms. Transform coding algorithms offer the advantage of selectable compression, limited only by the output image quality requirements; DPCM algorithms are generally less flexible in terms of achievable compression.

DPCM compression algorithms employ predictive coding to achieve compression; the gray level value of each pixel is predicted based upon previously encoded pixel gray level values, and the difference between the predicted and actual value of the pixel is then quantized and encoded. This encoding is done in the direction of the scan, one pixel at a time; the effects of the quantizer and prediction errors are thus minimal, generally manifesting themselves as edge effects in image areas containing sharp gray level transitions.

Transform coding algorithms use a method of encoding images which is much different from that of the DPCM algorithms. When an image is encoded using a transform coding algorithm, the image is broken into small (NxN) (N is the size of the transform matrix used) sub-blocks of pixels which are individually transformed and quantized. The effects of the quantizer error can be seen as a "blocking" effect; when the quantizer error is significant, all of the pixels within the sub-block are affected.

4.0 CONCLUSIONS AND RECOMMENDATIONS

In analyzing the results presented in Section 3.0, several conclusions were drawn concerning the performances of the four transform coding algorithms simulated relative to each other and to the performances of several DPCM algorithms simulated in an earlier study. These conclusions, in turn, led to the formulation of a number of recommendations as to which direction future research into gray scale compression studies involving transform coding should be directed.

4.1 Conclusions

1. The conditional zonal coding algorithm is the least complex of the four transform algorithms which were evaluated, but is more complex than the DPCM algorithms discussed. The image quality it achieved was very good for bit rates above 1 bit/pixel. At lower bit rates, however, the indiscriminate quantization employed by this technique caused significant distortion in the output images. The advantages offered by conditional zonal coding include low complexity, selectable compression, and reasonably good image quality at moderately low bit rates.

2. The adaptive zonal coding algorithm was slightly more complex than the conditional approach, but achieved much

better image quality. This was due to the algorithm's ability to adapt to image content. Adaptive zonal coding requires just one pass over the image to encode it, yet its performance was comparable to that of the Chen-Smith techniques, which require two passes. The advantages offered by adaptive zonal coding include moderately low complexity, selectable compression, and good image quality at low bit rates.

3. The basic Chen-Smith coding algorithm achieved excellent image quality at bit rates above 1 bit/pixel and very good image quality at bit rates as low as 0.5 bits/pixel. This algorithm, however, is very complex; it requires two passes over the image in order to encode it and requires a significant amount of statistical computations. The basic Chen-Smith coding technique offers the advantages of selectable compression and excellent image quality, but is relatively complex to implement.

4. The image dependent Chen Smith coding algorithm achieved the best image quality of all of the algorithms evaluated in this study, producing very good image quality at bit rates as low as 0.22 bits/pixel. This approach is the most complex of the four transform coding techniques simulated. Applications in which the use of the image dependent Chen-Smith coding technique would be advantageous include those

that require both high compression and excellent image quality without regard to system complexity.

5. At bit rates of 1 bit/pixel and above, the images produced in the DPCM simulations were virtually indistinguishable from the images produced in the transform coding simulations. In applications that require bit rates on the order of 1 to 1.5 bits/pixel, DPCM compression techniques would be more advantageous than transform coding techniques because they are less complex to implement.

6. The DPCM compression techniques did not produce bit rates below 0.63 bits/pixel; therefore, comparisons between the transform coding and DPCM algorithms could not be performed at the lower bit rates achieved in several of the transform coding simulations (0.1-0.3 bits/pixel). In applications where compression is more important than image quality, transform coding techniques have a distinct advantage over DPCM techniques.

7. Because the transform coding techniques offer the advantage of selectable compression, transform coding algorithms would be more favorable than DPCM algorithms in applications in which variable compression rates are required.

4.2 Recommendations for Further Study

1. A different image dependent variation of the Chen-Smith algorithm should be investigated. This variation should include an AC energy oriented sub-block classification method where standard AC energy thresholds are calculated and used to assign high or low classifications to sub-blocks in an image.

2. Optimization of the look-ahead technique used in the adaptive zonal coding technique should be performed in order to improve the image quality produced by this algorithm. It may be possible to improve the output image quality of this algorithm to the point where it makes the added complexity of the Chen-Smith algorithms unfavorable in some applications.

References

1. Chen, W., and Smith, C. "Adaptive Coding of Monochrome and Colour Images." IEEE Transactions on Communications, volume COM-25, no. 11, November 1977,pp.1285-1292.

2. Lloyd, S. P. "Least Squares Quantization in PCM." IEEE Transactions on Information Theory, volume IT-28, no. 2, March 1982,pp. 129-137.

3. Max, J. "Quantizing for Minimum Distortion." IRE Transactions on Information Theory, volume IT-6, no. 1, March 1960, pp. 7-12.

4. Delta Information Systems, "Computer Simulation of Gray Scale Compression Technique for Group 4 Facsimile.",Contract no. DCA100-83-C-0047, Task Order no. 84-002, May 1986.

APPENDIX A

SOFTWARE MANUAL

Table of Contents

Table of Contents (con't)

A.1 Operating Instructions

The Chen-Smith programs are run in three parts as diagrammed in figures A.1 and A.2. First, either GNSTIN or GNSTDP is run on an image file to gather statistics. These programs generate the variance matrix, which is used in program BITALL to allocate coding bits to the individual transform elements and in program MAXTRN to allow for the individual transform elements to have unit variance. The statistics generating programs also generate the class map which shows the activity level of each transformed sub-block, and the mean of the DC coefficients which is used in program MAXTRN for quantization. Second, program BITALL is run with an adjustable parameter to achieve the desired bit rate. BITALL allocates bits for each of the four classes created in the statistics generating program. Third, program MAXTRN is run with the variance matrix, the class map, the four bit maps and the Lloyd-Max (ref. 2,3) quantization levels as inputs. MAXTRN compresses, decompresses and writes the output image to file.

The conditional zonal program, ZNLTRN, is run with a statistics file as one of two inputs. The statistics file consists of an ordering sequence and corresponding dividing factors which are used in quantization. The second input is an adjustable parameter for the number of coefficients kept in quantization. ZNLTRN compresses, decompresses and writes the output image to file. The adaptive zonal program, THRTRN is run with the same statistics file mentioned above as one of two inputs. The second input is a adjustable threshold value used in

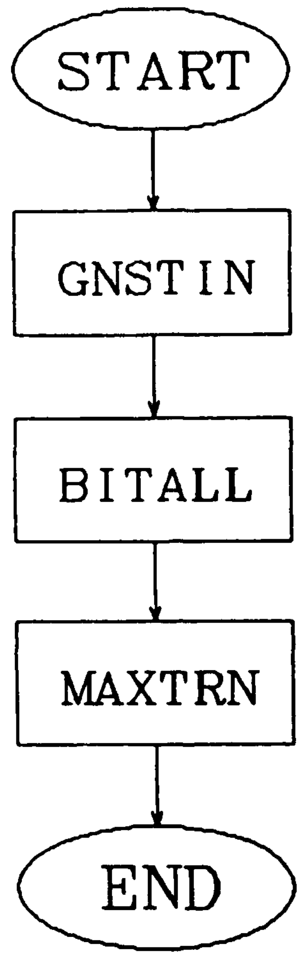

FIGURE A.1 DCT Independent Chen-Smith Data Flow Diagram

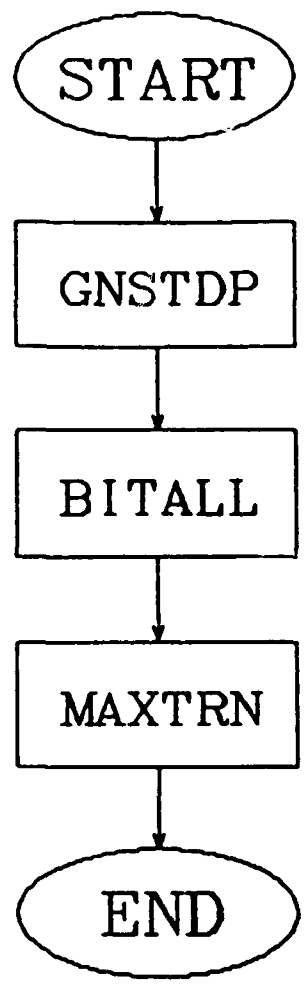

FIGURE A.2 DCT Dependent Chen-Smith Data Flow Diagram

quantization. THRTRN compresses, decompresses and writes the output image to file.

A.2 Software Documentation

The software documentation for the Discrete Cosine Transform programs is presented in this section, including structure charts, Nassi-Scneiderman flow charts for the software modules, and descriptions of the functions associated with the DCT programs.

A.2.1 Chen-Smith Coding Techniques

A.2.1.1 Statistics Generating Modules

A.2.1.1.1 Module GNSTIN

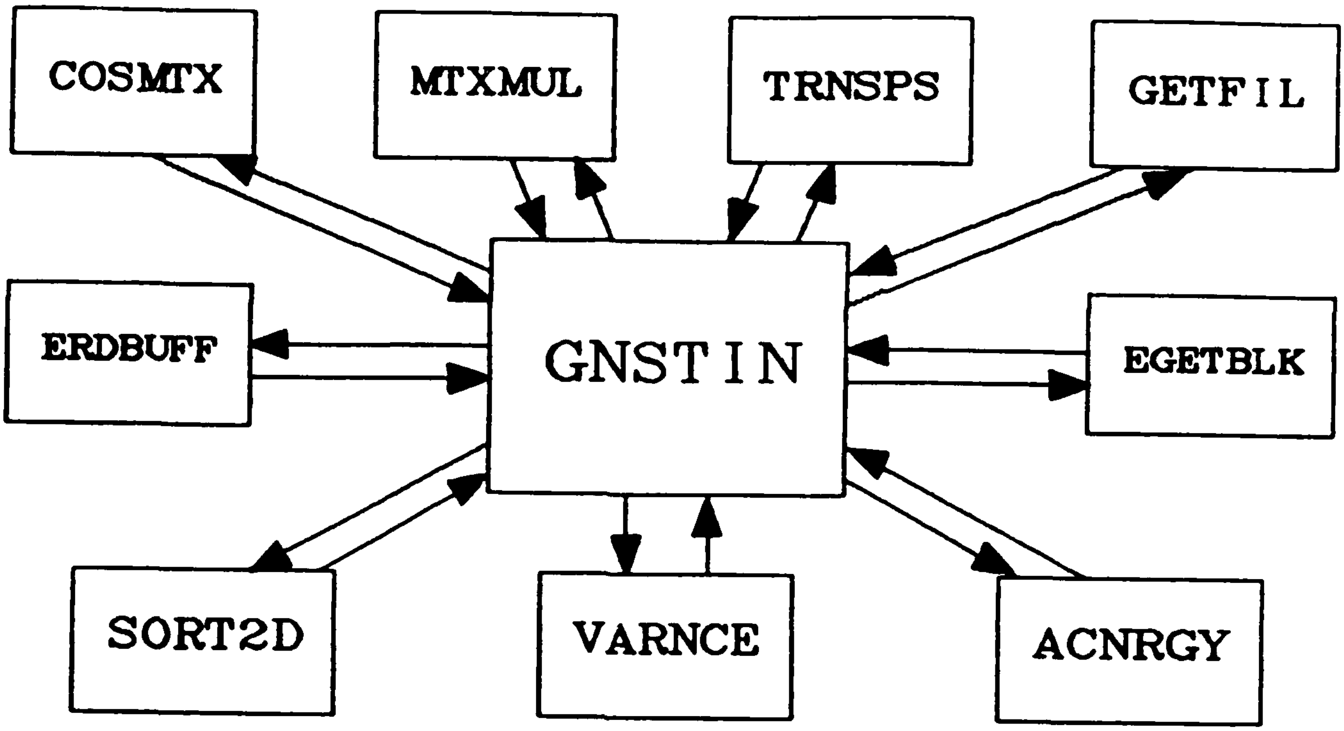

FIGURE A.3 Structure Chart for Module: GNSTIN

GNSTIN.FTN - Discrete Cosine Transform Image
Independent Statistics Generator Program

Open files and read input parameters

Calculate Cosine Matrix

Calculate Transpose of Cosine Matrix

Define the boundaries of the image file

Initialize count array

Do for the number of vertical sub-blocks

- Get row of horizontal sub-blocks
- Do for the number of horizontal sub-blocks
  - Get sub-block and transform the binary numbers of a sub-block of the image file to real numbers
  - Perform matrix multiplications to transform sub-block matrix
  - Calculate AC energy in sub-block
  - Sum up the squares of the AC coefficients for calculation of the variance matrix and sum up the DC coefficient for the calculation of its mean

Calculate the number of sub-blocks and the DC coefficient mean

Calculate the DC coefficient variance

GNSTIN.FTN - Discrete Cosine Transform Image Independent Statistics Generator Program

- Do for the number of vertical sub-blocks
  - Do for the number of horizontal sub-blocks
    - Put the AC energies of the image into an array
- Do for the number of vertical pixels
  - Do for the number of horizontal pixels
    - Calculate the AC coefficients variance to complete the variance matrix
- Sort energies of each sub-block
- Calculate the sub-block classifications
- Do for number of vertical sub-blocks
  - Do for number of horizontal sub-blocks
    - Classify each sub-block according to non-uniform bounds
- Do for number of vertical pixels
  - Do for number of horizontal pixels
    - Write the variance matrix to file
- Do for number of vertical sub-blocks
  - Do for number of horizontal sub-blocks
    - Write the class map matrix to file

GNSTIN.FTN - Discrete Cosine Transform Image Independent Statistics Generator Program

Write to file the mean of the DC coefficient
Close files
E N D

ERDBUFF.FTN - Buffer Reading Subroutine

Move down to proper spot reading point of Image file
Do for the number of vertical dimension of sub-blocks   Read horizontal block of pixels
Return
E N D

COSMTX.FTN - Cosine Matrix Subroutine

- Do for the number of vertical rows of pixels
  - Is this the first vertical row of pixels?
    - YES: Make the multiplying coefficient equal to "1/sqrt(2)"
    - NO: Make the multiplying coefficient equal to "1"
  - Do for number of horizontal pixels
    - Calculate the cosine coefficient
- Return
- E N D

TRNSPS.FTN - Transpose Subroutine

```
Do for the number of vertical row of pixels
    Do for the number of horizontal pixels
        The transform matrix is equal to the computed cosine matrix
Return
E N D
```

EGETBLK.FTN - Sub-block Retrieving Subroutine

Move across to proper spot in buffer

Do for the number of vertical dimension of sub-blocks

> Do for half the horizontal dimension of sub-blocks
>
> > Do two times
> >
> > > Take half a word which is one pixel from the buffer

Return

E N D

SORT2D.FTN - Sorting Subroutine

Do for I equals one to dimension minus one of array

Do for dimension of array down to (I + 1) in steps of (-1)

Is the adjacent pair of array elements out of order? YES	NO
Exchange the pair of array elements	Leave the array elements as they are

Return

E N D

MTXMUL.FTN - Matrix Multiplication Subroutine

Do for the number of vertical dimension of sub-blocks

Do for the number of horizontal dimension of sub-blocks

Do multiplication of pixels from MATRIX"1" & MATRIX"2"

Should the result of the matrix multiplication be put in MATRIX"1" ?

YES

Do for vertical dimension of sub-blocks

Do for horizontal dimension of sub-blocks

Put result in MATRIX"1"

NO

Do for vertical dimension of sub-blocks

Do for horizontal dimension of sub-blocks

Put result in MATRIX"2"

Return

E N D

ACNRGY.FTN - AC Energy Calculation Function

Do for number of vertical pixels

Do for number of horizontal pixels

Sum up the AC energies of each pixel position

ACNRGY equals the total of all AC energies

Return

E N D

VARNCE.FTN - Variance Subroutine

```
Do for number of vertical pixels
    Do for number of horizontal pixels
        Sum up the squares of the AC coefficients
Return
E N D
```

Program Documentation for module: GNSTIN

PROGRAM: GNSTIN

DESCRIPTION: This program uses an adaptive coding technique known as the Discrete Cosine Transform (DCT) and follows the Chen-Smith (ref. 1) coding algorithm. Transformed blocks are sorted into classes by the level of image activity. Within each activity level, coding bits are allocated to individual transform elements according to the variance matrix of the transformed data. An equal amount of blocks will be distributed in each class independent of excessively high or image activity. This program generates the variance matrix which is used in module BITALL to allocate coding bits to individual transform elements and module MAXTRN to make the individual transform elements have unit variance. This program also generates, the class map which shows the activity level of each transformed sub-block and the mean of the DC coefficient which is used in module MAXTRN in quantization.

RUNSTRING: GNSTIN,<INPUT NAME>,<OUTPUT NAME>

INPUT NAME — Input image file name

OUTPUT NAME — Output statistics file

ORDER OF INPUT PARAMETERS:

1) Dimension of sub-blocks

2) Number of sub-block classification levels

MODULES CALLED:

ERDBUFF — Subroutine to read a horizontal line of the input image into the FTN77 buffer.

COSMTX — Subroutine to put in memory the cosine matrix.

Program Documentation for module: GNSTIN

TRNSPS	Subroutine to put in memory the transpose of the cosine matrix.
EGETBLK	Subroutine to retrieve a block of data from the FTN77 buffer.
SORT2D	Subroutine to sort an array of AC energies.
MTXMUL	Subroutine to do matrix multiplications of real numbers.
GETFIL	Subroutine to open input image an file for processing.
ACNRGY	Subroutine to calculate AC energy of sub-blocks
VARNCE	Subroutine to add the squares of the AC coefficients for calculation of the variance matrix.

NAMED COMMON
DESCRIPTIONS:

Block Name: GFMBLK
Module Common to: RDBUFF

Descriptions:

IMGFIL	Input image file name
EXISTS	File exists flag
ISTAT	File status variable
RECLEN	Record length in bytes
NUMREC	Number of records in input file
RECRDS	Number of records in primary file
FTN77	Fortran read buffer
TEMBUF	Temporary read buffer
ACCTYP	File access flag

Program Documentation for module: GNSTIN

Block Name:	GTBLK
Module Common to:	RDBUFF,GETBLK,SORT2D

Descriptions:

OUTBUF	Output buffer
ACSORT	Array holding sorted AC energies
ACMTX	Matrix holding AC energies

Subroutine Documentation for module: ERDBUFF

SUBROUTINE: ERDBUFF

MODULES CALLED FROM: GNSTIN, GNSTDP

PURPOSE: This subroutine reads a horizontal sub-block of data from the image file into the FTN77 buffer.

MODULES CALLED:

LGBUF	Subroutine to make the buffer size larger.

CALLING FORMAT: CALL ERDBUFF(YDIM,YVAL,INLU)

ARGUMENT DESCRIPTIONS:

YDIM	Y dimension of the buffer in words
YVAL	Y coordinate of file for reading
INLU	LU for the input image file

NAMED COMMON DESCRIPTIONS:

Block Name: GFMBLK
Module Common to: GNSTIN, GNSTDP

Descriptions:

IMGFIL	Input image file name
EXISTS	File exists flag
ISTAT	File status variable
RECLEN	Record length in bytes
NUMREC	Number of records in input file

Subroutine Documentation for module: ERDBUFF

RECRDS	Number of records in primary file
FTN77	Fortran read buffer
TEMBUF	Temporary read buffer
ACCTYP	File access flag

Block Name: GTBLK
Module Common to: GNSTIN, GNSTDP, EGETBLK, SORT2D

Description:

OUTBUF	Output buffer
ACSORT	Array holding sorted AC energies
ACMTX	Matrix holding AC energies

Subroutine Documentation for module: COSMTX

SUBROUTINE: COSMTX

MODULES CALLED FROM: MAXTRN, THRTRN, ZNLTRN, GNSTIN, GNSTDP

PURPOSE: This subroutine creates the cosine matrix

CALLING FORMAT: CALL COSMTX(XFORM,MTXDIM)

ARGUMENT DESCRIPTIONS:

XFORM	Transform matrix to be computed
MTXDIM	Dimension of transform matrix

Subroutine Documentation for module: TRNSPS

SUBROUTINE: TRNSPS

MODULES CALLED FROM: MAXTRN, THRTRN, ZNLTRN, GNSTIN, GNSTDP

PURPOSE: This subroutine puts the transpose of the cosine matrix in TRXFORM

CALLING FORMAT: CALL TRNSPS(XFORM,TRXFORM,MTXDIM)

ARGUMENT DESCRIPTIONS:

XFORM	Transform matrix COSMTX
TRXFORM	Transpose of the transform matrix
MTXDIM	Matrix dimension

Subroutine Documentation for module: EGETBLK

SUBROUTINE: EGETBLK

MODULES CALLED FROM: GNSTIN, GNSTDP

PURPOSE: This subroutine retrieves a block of data from the block buffer and places it in the transform data buffer for transformation.

CALLING FORMAT: CALL EGETBLK(XVAL,YVAL,XSIZ,YSIZ,BLKNAM)

ARGUMENT DESCRIPTIONS:

XVAL	Upper left X file coordinate
YVAL	Upper left Y file coordinate
XSIZ	X Block dimension
YSIZ	Y Block dimension
BLKNAM	Memory to hold a block of data to be retrieved

NAMED COMMON DESCRIPTIONS:

Block Name: GTBLK
Module Common to: GNSTIN, GNSTDP, ERDBUFF, SORT2D

Description:

OUTBUF	Output buffer
ACSORT	Array holding sorted AC energies
ACMTX	Matrix holding AC energies

Subroutine Documentation for module: SORT2D

SUBROUTINE: SORT2D

MODULE CALLED FROM: GNSTIN, GNSTDP

PURPOSE: This subroutine sorts an array of AC energies to get appropriate class bounds for the class map.

CALLING FORMAT: CALL SORT2D(DIMNSN)

ARGUMENT DESCRIPTIONS:

DIMNSN   Dimension of the array to be sorted

NAMED COMMON DESCRIPTIONS:

Block Name: GTBLK
Module Common to: ERDBUFF, EGETBLK, GNSTIN, GNSTDP

Description:

OUTBUF	Output buffer
ACSORT	Array holding sorted AC energies
ACMTX	Matrix holding AC energies

Subroutine Documentation for module: MTXMUL

SUBROUTINE: MTXMUL

MODULES CALLED FROM: MAXTRN, THRTRN, ZNLTRN, GNSTIN, GNSTDP

PURPOSE: This subroutine will do matrix multiplications of real numbers on two matrices.

CALLING FORMAT: CALL MTXMUL(MTX1,MTX2,SIZE,DEST)

ARGUMENT DESCRIPTIONS:

MTX1	Matrix one, ordering is important
MTX2	Matrix two, again ordering is important
SIZE	Size of matrices
DEST	Destination of the result of (MTX1 * MTX2) (1 Result places in MTX1, 2 Result in MTX2)

Function Documentation for module: ACNRGY

Function: ACNRGY

MODULES CALLED FROM: GNSTIN, GNSTDP

PURPOSE: This function calculates the AC energy of a sub-block. The AC energies are then used for block classification.

CALLING FORMAT: X = ACNRGY(MTX,XDIM,YDIM)

ARGUMENT DESCRIPTIONS:

MTX	Data matrix
XDIM	X dimension of data matrix
YDIM	Y dimension of data matrix

Subroutine Documentation for module: VARNCE

SUBROUTINE: VARNCE

MODULES CALLED FROM: GNSTIN, GNSTDP

PURPOSE: This subroutine adds the squares of the AC coefficients within the sub-block over the entire image. The sums of the squares will be later used to calculate the variance matrix.

CALLING FORMAT: CALL VARNCE(TRMTX,VARMTX,XDIM,YDIM)

ARGUMENT DESCRIPTIONS:

TRMTX	Input transformed matrix
VARMTX	Output variance matrix
XDIM	X dimension of the transformed matrix
YDIM	Y dimension of the transformed matrix

A.2.1.1.2 Module GNSTDP

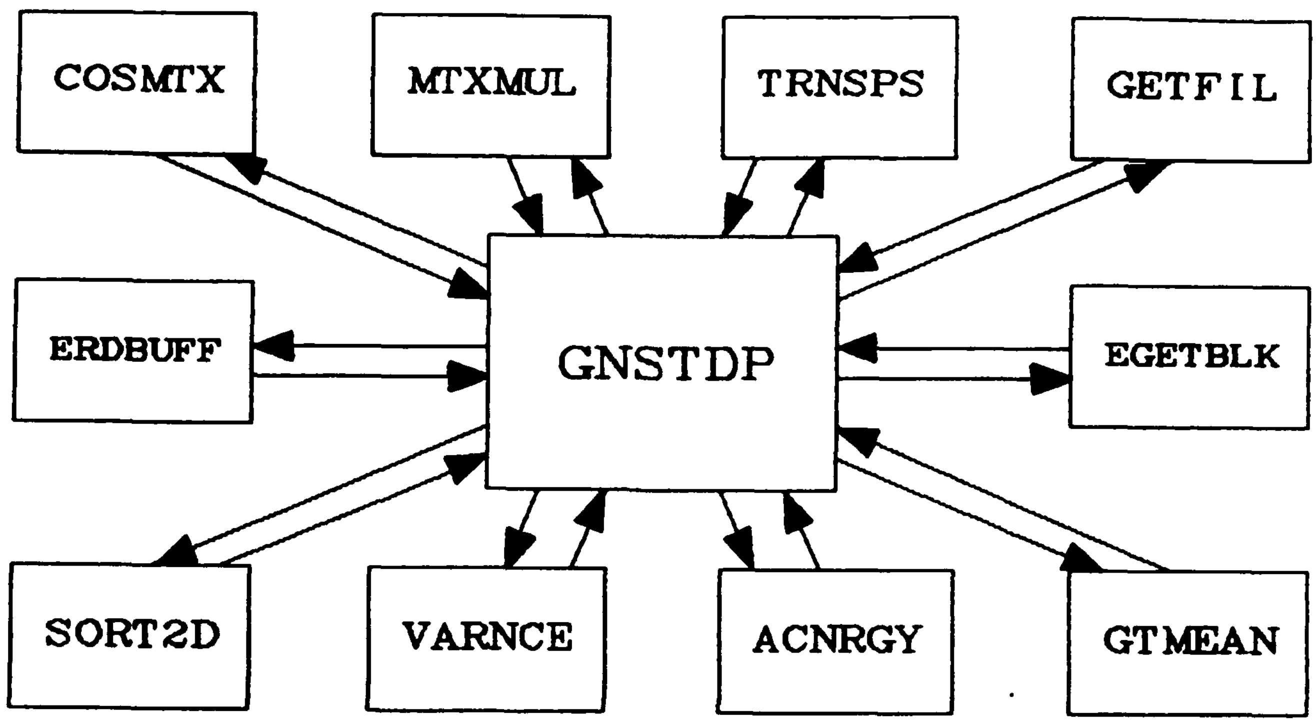

FIGURE A.4 Structure Chart for Module: GNSTDP

GNSTDP.FTN - Discrete Cosine Transform Image
Dependent Statistics Generator Program

- Open files and read input parameters
- Calculate Cosine Matrix
- Calculate Transpose of Cosine Matrix
- Define the boundaries of the image file
- Initialize count array
- Do for the number of vertical sub-blocks
  - Get row of horizontal sub-blocks
  - Do for the number of horizontal sub-blocks
    - Get sub-block and transform the binary numbers of a sub-block of the image file to real numbers
    - Perform matrix multiplications to transform sub-block matrix
    - Calculate AC energy in sub-block
    - Sum up the squares of the AC coefficients for calculation of the variance matrix and sum up the DC coefficients for the calculation of its mean
- Calculate the number of sub-blocks and the DC coefficient mean
- Calculate the DC coefficient variance

GNSTDP.FTN - Discrete Cosine Transform Image Dependent Statistics Generator Program

Do for number the of vertical sub-blocks

- Do for the number of horizontal sub-blocks
  - Put the AC energies of the image into a array

Do for the number of vertical pixels

- Do for the number of horizontal pixels
  - Calculate the AC coefficients variance to complete the variance matrix

Sort energies of each sub-block

Calculate the mean of the sorted array

Let class bound two equal the mean of the sorted array

Calculate the mean of the sorted array up to class bound two and let that mean equal class bound one

Calculate the mean of the sorted array after class bound two

Let the mean of the sorted array after class bound two equal class bound three

Calculate the sub-block classifications

Do for number of vertical sub-blocks

- Do for number of horizontal sub-blocks
  - Classify each sub-block according to non-uniform bounds

GNSTDP.FTN - Discrete Cosine Transform Image Dependent Statistics Generator Program

- Do for number of vertical pixels
  - Do for number of horizontal pixels
    - Write the variance matrix to file
- Do for number of vertical sub-blocks
  - Do for number of horizontal sub-blocks
    - Write the class map matrix to file
- Write to file the mean of the DC coefficient
- Close files
- E N D

GTMEAN.FTN - Mean Calculation Function

Do for the number in the array

Sum up the coefficients in the array

Calculate the mean of the array and set it equal to GTMEAN

Return

E N D

Program Documentation for module: GNSTDP

PROGRAM: GNSTDP

DESCRIPTION: This program uses an adaptive coding technique known as the Discrete Cosine Transform (DCT) and follows the Chen-Smith (ref. 1) coding algorithm. Transformed blocks are sorted into classes by the level of image activity. Within each activity level, coding bits are allocated to individual transform elements according to the variance matrix of the transformed data. This program is image dependent unlike module GNSTIN. The amount of blocks in each class will depend upon the image activity. This program generates the variance matrix which is used in module BITALL to allocate coding bits to individual transform elements and module MAXTRN to make the individual transform elements have unit variance. This program also generates, the class map which shows the activity level of each transformed sub-block and the mean of the DC coefficient which is used in module MAXTRN in quantization.

RUNSTRING: GNSTDP,<INPUT NAME>,<OUTPUT NAME>

- INPUT NAME — Input image file name
- OUTPUT NAME — Output statistics file

ORDER OF INPUT PARAMETERS:

1) Dimension of sub-blocks

2) Number of sub-block classification levels

MODULES CALLED:

- ERDBUFF — Subroutine to read a horizontal line of the input image into the FTN77 buffer.
- COSMTX — Subroutine to put in memory the cosine matrix.

Program Documentation for module: GNSTDP

TRNSPS	Subroutine to put in memory the transpose of the cosine matrix.
EGETBLK	Subroutine to retrieve a block of data from the FTN77 buffer.
SORT2D	Subroutine to sort an array of ac energies.
MTXMUL	Subroutine to do matrix multiplications of real numbers.
GTMEAN	Function to calculate the mean of an array. Used to determine class bounds.
ACNRGY	Function to calculate AC energy of sub-blocks.
VARNCE	Subroutine to add the squares of the AC coefficients for calculation of the variance matrix.
GETFIL	Subroutine to open input an image file for processing.

NAMED COMMON
DESCRIPTIONS:

Block Name: GFMBLK
Module Common to: RDBUFF

Descriptions:

IMGFIL	Input image file name
EXISTS	File exists flag
ISTAT	File status variable
RECLEN	Record length in bytes
NUMREC	Number of records in input file
RECRDS	Number of records in primary file
FTN77	Fortran read buffer

Program Documentation for module: GNSTDP

TEMBUF	Temporary read buffer
ACCTYP	File access flag

Block Name: GTBLK
Module Common to: RDBUFF,GETBLK,SORT2D

Descriptions:

OUTBUF	Output buffer
ACSORT	Array holding sorted AC energies
ACMTX	Matrix holding AC energies

Function Documentation for module: GTMEAN

Function: GTMEAN

MODULE CALLED FROM: GNSTDP

PURPOSE: This function calculates mean of an array.

CALLING FORMAT: X = GTMEAN(FIRST,LAST)

ARGUMENT DESCRIPTIONS:

FIRST The starting point of the calculation.

LAST The ending point of the calculation.

A.2.1.2 BIT ALLOCATING PROGRAM: BITALL

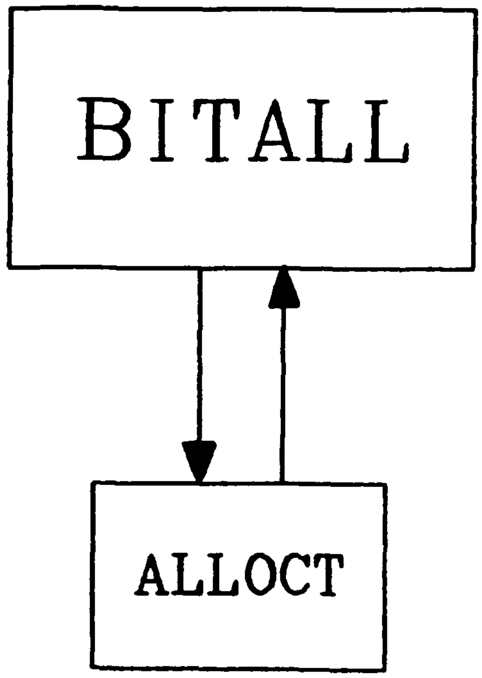

FIGURE A.5 Structure Chart for Module: BITALL

BITALL.FTN - Discrete Cosine Transform
Bit Allocation Program

- Open files and read input parameters
- Do for number of vertical pixels
  - Do for number of horizontal pixels
    - Read in the current pixel of the variance matrix
- Calculate the desired bit targets used in determining the bit maps
- Do for number of classes
  - Do while current bit is less than target
    - Allocate bits for current bit map
  - Do for number of vertical pixels
    - Do for number of horizontal pixels
      - Write to file the current bit map
- Write the average number of bits per pixel
- Close files
- E N D

ALLOCT.FTN - Bit Allocating Function

Do for dimension of matrix

Do for dimension of matrix

Is this an AC coefficient?

YES	NO
Calculate bits to allocate for the AC coefficient	Allocate eight bits for the DC coefficient

Return the average bits per pixel

E N D

Program Documentation for module: BITALL

PROGRAM: BITALL

DESCRIPTION: This program will allocate bits for each of the four classes created in the statistics generating program. The coding bits are allocated to individual transform elements according to the variance matrix of of the transformed data. Bits are then distributed between "busy" and "quiet" image areas to provide the desired adaptivity; more bits assigned to the areas of high image activity and fewer bits assigned to those of low activity.

RUNSTRING: BITALL,<INPUT NAME>,<OUTPUT NAME>

INPUT NAME — Input variance matrix

OUTPUT NAME — Output bit allocation maps

ORDER OF **INPUT PARAMETERS:**

1) Dimension of sub-blocks

2) Desired number of coded bits per pixel

MODULES CALLED:

ALLOCT — Function that creates the bit allocation matrices for the different classes of the classification map.

Function Documentation for module: ALLOCT

Function: ALLOCT

MODULE CALLED FROM: BITALL

PURPOSE: This function calculates the bit allocation matrices for the different classes of the classification map.

CALLING FORMAT: X = ALLOCT(BAMTX,VARMTX,PARM,XDIM)

ARGUMENT DESCRIPTIONS:

BAMTX	Output bit allocation matrix
VARMTX	Input variance matrix
PARM	Input parameter to the bit allocation function
XDIM	Matrix dimension

A.2.1.3 CODING PROGRAM: MAXTRN

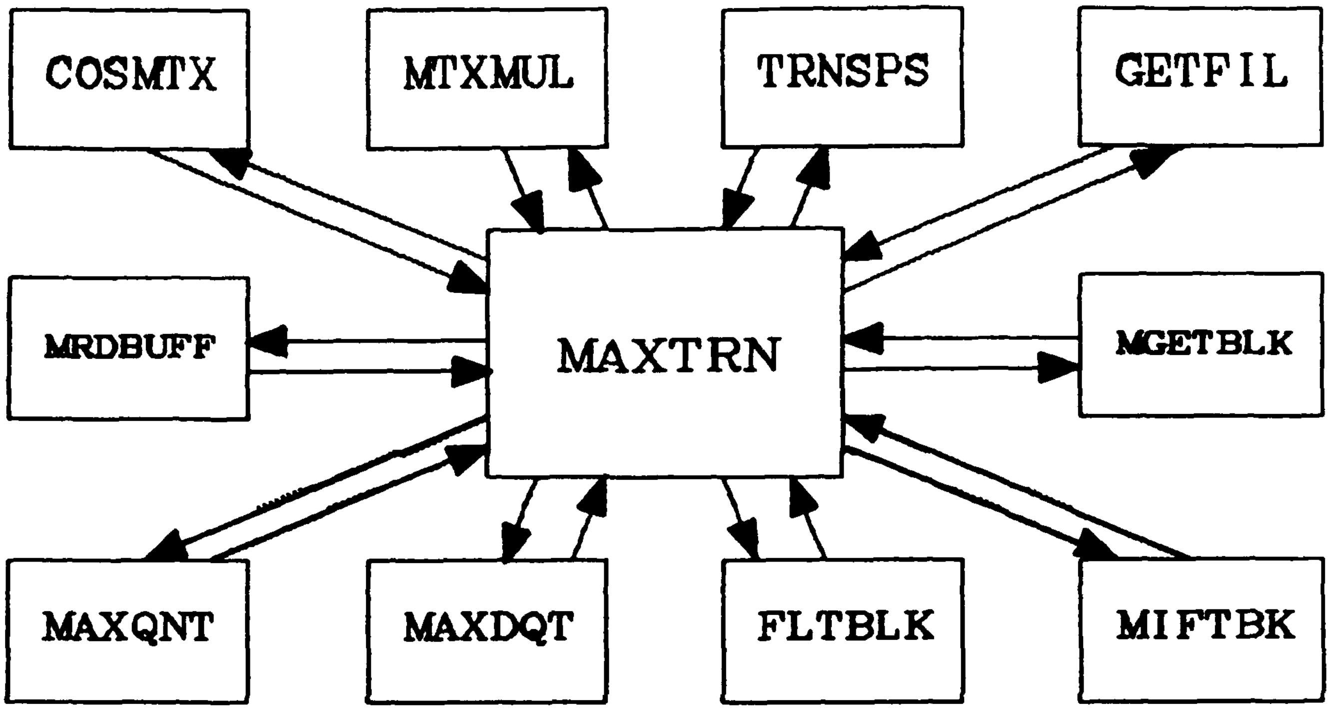

FIGURE A.6 Structure Chart for Module: MAXTRN

MAXTRN.FTN - Discrete Cosine Transform Program
using the Lloyd-Max Quantizing Method

Open files and read input parameters

Do for the number of vertical pixels

- Do for the number of horizontal pixels
  - Read in the current pixel of the variance matrix that was created in the Statistics Generator

Do for the number of vertical sub-blocks

- Do for the number of horizontal sub-blocks
  - Read the class map that was created in the Statistics Generator

Read the mean of the DC coefficient created in the Statistics Generator

Do for the number of vertical pixels

- Do for the number of horizontal pixels
  - Read in bit maps that were created in module BITALL

Do for the current number of levels

- Read in the current Lloyd-Max quantization levels

Calculate Cosine Matrix

Calculate Transpose of Cosine Matrix

Define the boundaries of the image file

Initialize total number of bits

MAXTRN.FTN - Discrete Cosine Transform Program
using the Lloyd-Max Quantizing Method

- Do for the number of vertical sub-blocks
  - Get row of horizontal sub-blocks
  - Do for the number of horizontal sub-blocks
    - Get sub-block and transform the binary numbers of a sub-block of the image file to real numbers
    - Perform matrix multiplications to transform sub-block matrix
    - Quantize transformed sub-block
    - Perform an integer filtering process that puts back in range out of range coefficients due to quantization error
    - Dequantize transformed quantized sub-block
    - Perform matrix multiplications to transform the sub-block back to original form. (The sub-block will not be exactly the same due to quantization error)
    - Perform filtering process that puts back in range the out of range pixels that were due to quantization error

MAXTRN.FTN - Discrete Cosine Transform Program
using the Lloyd-Max Quantizing Method

Do for the number of vertical pixels

Do for the number of horizontal pixels

Do for the number of bits per word

Pack bits into word

Increment total pixels being processed

Write a row of horizontal sub-blocks to output file

Calculate and print out compression statistics

E N D

MRDBUFF.FTN - Buffer Reading Subroutine

Move down to proper spot reading point of Image file
Do for the number of vertical dimension of sub-blocks   Read horizontal block of pixels
Return
E N D

MGETBLK.FTN - Sub-block Retrieving Subroutine

```
Move across to proper spot in buffer
Do for the number of vertical dimension of sub-blocks
    Do for half the horizontal dimension of sub-blocks
        Do two times
            Take half a word which is one pixel from the buffer
Return
E N D
```

MAXQNT.FTN - Lloyd-Max Quantizing Subroutine

- Initialize quantizing sub-block
- Do for number of vertical pixels
  - Do for number of horizontal pixels
    - Determine how many levels to use according to the position MAXTRN is at on the image map and keep track of bits that are used
    - Divide the current coefficient by the standard deviation to make the coefficient have unit variance
    - Put the quantization levels in an array
    - Is this the DC coefficient?
      - YES: Let the mean of the DC coefficient divided by its standard deviation be the center of the quantization levels
      - NO: Let zero be the center of the quantization levels
    - Determine the correct quantization level
    - Put the correct quantization level in the output matrix
- Return
- E N D

MAXDQT.FTN - Lloyd-Max Dequantizing Subroutine

Initialize dequantizing sub-block

Do for number of vertical pixels

Do for number of horizontal pixels

Determine how many levels to use according to the position MAXTRN is at on the image map

Put the dequantization levels in an array

Is this the DC coefficient?

YES

Let the mean of the DC coefficient divided by its standard deviation be the center of the dequantization levels

NO

Let zero be the center of the dequantization levels

Determine the correct dequantization level

Put the correct dequantization level in the output matrix

Return

E N D

FLTBLK.FTN - Filtering Subroutine

Do for vertical dimension of sub-block

Do for horizontal dimension of sub-block

Is current pixel of sub-block less than "0"?

YES

Make the current pixel of sub-block equal to "0"

Is current pixel of sub-block greater than "255"?

YES

Make the current pixel of sub-block equal to "255"

Return

E N D

MIFTBK.FTN - Filtering Subroutine

Do for vertical dimension of sub-block

- Do for horizontal dimension of sub-block
  - Is current pixel of sub-block less than "0"?
    - YES
      - Make the current pixel of sub-block equal to "0"
  - Is current pixel of sub-block greater than "255"?
    - YES
      - Make the current pixel of sub-block equal to "255"

Return

E N D

Program Documentation for module: MAXTRN

PROGRAM: MAXTRN

DESCRIPTION: This program uses an adaptive coding technique known as the Discrete Cosine Transform (DCT) and follows the Chen-Smith (ref. 1) algorithm. The program interactively inquires for, then accepts input parameters for the dimensions of the sub-blocks. The quantization method used in this program is the Lloyd-Max optimal quantization scheme (ref. 2,3) with the probability density and transform sample modeled as equation 3.1. This coding method will be applied to three images, a face photo, an aerial photo and a crowd scene. A summary of each run is printed including compression statistics and RMS values.

RUNSTRING:

```
MAXTRN,<INPUT NAME>,<OUTPUT NAME>,<STAT FILE>,
       <BIT MAPS>,<256 LEVELS>,<128 LEVELS>,
       <64 LEVELS>,<32 LEVELS>
```

INPUT NAME	Input image file name
OUTPUT NAME	Output reconstructed image file name
STAT FILE	The variance matrix and image map from the statistics generating program
BIT MAPS	The four bit maps created in BITALL
256 LEVELS	256 of the Lloyd-Max quantization levels
128 LEVELS	128 of the Lloyd-Max quantization levels
64 LEVELS	64 of the Lloyd-Max quantization levels
32 LEVELS	32 of the Lloyd-Max quantization levels

MODULES CALLED:

MRDBUFF	Subroutine to read a horizontal line of the input image into the FTN77 buffer.
COSMTX	Subroutine to put in memory the cosine matrix.

Program Documentation for module: MAXTRN

TRNSPS	Subroutine to put in memory the transpose of the cosine matrix.
MTXMUL	Subroutine to do matrix multiplications of real numbers
MGETBLK	Subroutine to retrieve a block of data from the FTN77 buffer.
MAXQNT	Subroutine to quantize blocks of data.
MAXDQT	Subroutine to dequantize blocks of data.
FLTBLK	Subroutine to filter out, out of range real pixels.
MIFTBK	Subroutine to filter out, out of range integer coefficients.
GETFIL	Subroutine to open input image an file for processing.

NAMED COMMON
DESCRIPTIONS:

Block Name: GFMBLK
Module Common to: MRDBUFF

Descriptions:

IMGFIL	Input image file name
EXISTS	File exists flag
ISTAT	File status variable
RECLEN	Record length in bytes
NUMREC	Number of records in input file
RECRDS	Number of records in primary file
FTN77	Fortran read buffer
TEMBUF	Temporary read buffer

Program Documentation for module: MAXTRN

ACCTYP	File access flag

Block Name: GTBLK
Module Common to: MRDBUFF,MGETBLK

Descriptions:

OUTBUF	Output buffer

Subroutine Documentation for module: MRDBUFF

SUBROUTINE:	MRDBUFF
MODULE CALLED FROM:	MAXTRN
PURPOSE:	This subroutine reads a horizontal sub-block of data from the image file into the FTN77 buffer.

MODULES CALLED:

LGBUF	Subroutine to make the buffer size larger.

CALLING FORMAT: CALL MRDBUFF(YDIM,YVAL,INLU)

ARGUMENT DESCRIPTIONS:

YDIM	Y dimension of the buffer in words
YVAL	Y coordinate of file for reading
INLU	LU for the input image file

NAMED COMMON DESCRIPTIONS:

Block Name: GFMBLK
Module Common to: MAXTRN

Descriptions:

IMGFIL	Input image file name
EXISTS	File exists flag
ISTAT	File status variable
RECLEN	Record length in bytes
NUMREC	Number of records in input file

Subroutine Documentation for module: MRDBUFF

RECRDS	Number of records in primary file
FTN77	Fortran read buffer
TEMBUF	Temporary read buffer
ACCTYP	File access flag

Block Name: GTBLK
Module Common to: MGETBLK,MAXTRN

Description:

OUTBUF	Output buffer

Subroutine Documentation for module: MGETBLK

SUBROUTINE:	MGETBLK
MODULE CALLED FROM:	MAXTRN
PURPOSE:	This subroutine retrieves a block of data from the block buffer and places it in the transform data buffer for transformation.
CALLING FORMAT:	CALL MGETBLK(XVAL,YVAL,XSIZ,YSIZ,BLKNAM)

ARGUMENT DESCRIPTIONS:

XVAL	Upper left X file coordinate
YVAL	Upper left Y file coordinate
XSIZ	X Block dimension
YSIZ	Y Block dimension
BLKNAM	Memory to hold a block of data to be retrieved

NAMED COMMON DESCRIPTIONS:

Block Name: GTBLK
Module Common to: MGETBLK,MAXTRN

Description:

OUTBUF	Output buffer

Subroutine Documentation for module: MAXQNT

SUBROUTINE: MAXQNT

MODULE CALLED FROM: MAXTRN

PURPOSE: This subroutine uses the Lloyd-Max optimal quantization scheme (ref. 2,3). The quantization process takes the current coefficient and determines the correct quantization interval and represents the coefficient by the input level that corresponds to that interval. The quantization levels used are the ones that are described in the Lloyd-Max algorithm.

CALLING FORMAT:

```
CALL MAXQNT(QNTMTX,BLKBUF,IMGMAP,VARRY,TTLBTS,
            V1MAP,V2MAP,V3MAP,V4MAP,XDIM,YDIM,
            I,J,X256,X128,X64,X32)
```

ARGUMENT DESCRIPTIONS:

QNTMTX	Output quantized matrix
BLKBUF	Input matrix to be quantized
IMGMAP	The total image energy map
VARRY	The variance matrix
TTLBTS	Total actual bits sent
V1MAP	Variance map one
V2MAP	Variance map two
V3MAP	Variance map three
V4MAP	Variance map four
XDIM	X Dimension of inputted matrix
YDIM	Y Dimension of inputted matrix

Subroutine Documentation for module: MAXQNT

I,J	The current position of IMGMAP sent from MAXTRN
X256	256 Input levels
X128	128 Input levels
X64	64 Input levels
X32	32 Input levels

Subroutine Documentation for module: MAXDQT

SUBROUTINE: MAXDQT

MODULE CALLED FROM: MAXTRN

PURPOSE: This subroutine will dequantize a transformed quantized sub-block of pixels. The dequantization process takes the quantized pixel and dequantizes it by giving the pixel its corresponding output level.

CALLING FORMAT:

```
CALL MAXDQT(QNTMTX,BLKBUF,IMGMAP,VARRY,
            V1MAP,V2MAP,V3MAP,V4MAP,XDIM,YDIM,
            I,J,Y256,Y128,Y64,Y32)
```

ARGUMENT DESCRIPTIONS:

QNTMTX	Input quantized matrix
BLKBUF	Output matrix to be dequantized
IMGMAP	The total image energy map
VARRY	The variance matrix
V1MAP	Variance map one
V2MAP	Variance map two
V3MAP	Variance map three
V4MAP	Variance map four
XDIM	X Dimension of inputted matrix
YDIM	Y Dimension of inputted matrix
I,J	The current position of IMGMAP sent from MAXTRN
Y256	256 Output levels
Y128	128 Output levels
Y64	64 Output levels
Y32	32 Output levels

Subroutine Documentation for module: FLTBLK

SUBROUTINE:	FLTBLK
MODULES CALLED FROM:	MAXTRN, THRTRN, ZNLTRN
PURPOSE:	This subroutine will filter out reconstructed real pixels that are out of range. (eg. Larger than 255, or smaller than 0)
CALLING FORMAT:	CALL FLTBLK(MATRIX,DIM)

ARGUMENT DESCRIPTIONS:

MATRIX	Matrix to be filtered
DIM	Dimension of matrix to be filtered

Subroutine Documentation for module: MIFTBK

SUBROUTINE: MIFTBK

MODULE CALLED FROM: MAXTRN

PURPOSE: This subroutine will filter out reconstructed integer coefficients that are out of range. (eg. Larger than 255, or smaller than 0)

CALLING FORMAT: CALL MIFTBK(MATRIX,DIM)

ARGUMENT DESCRIPTIONS:

MATRIX	Matrix to be filtered
DIM	Dimension of matrix to be filtered

A.2.2.1 CODING PROGRAM: ZNLTRN

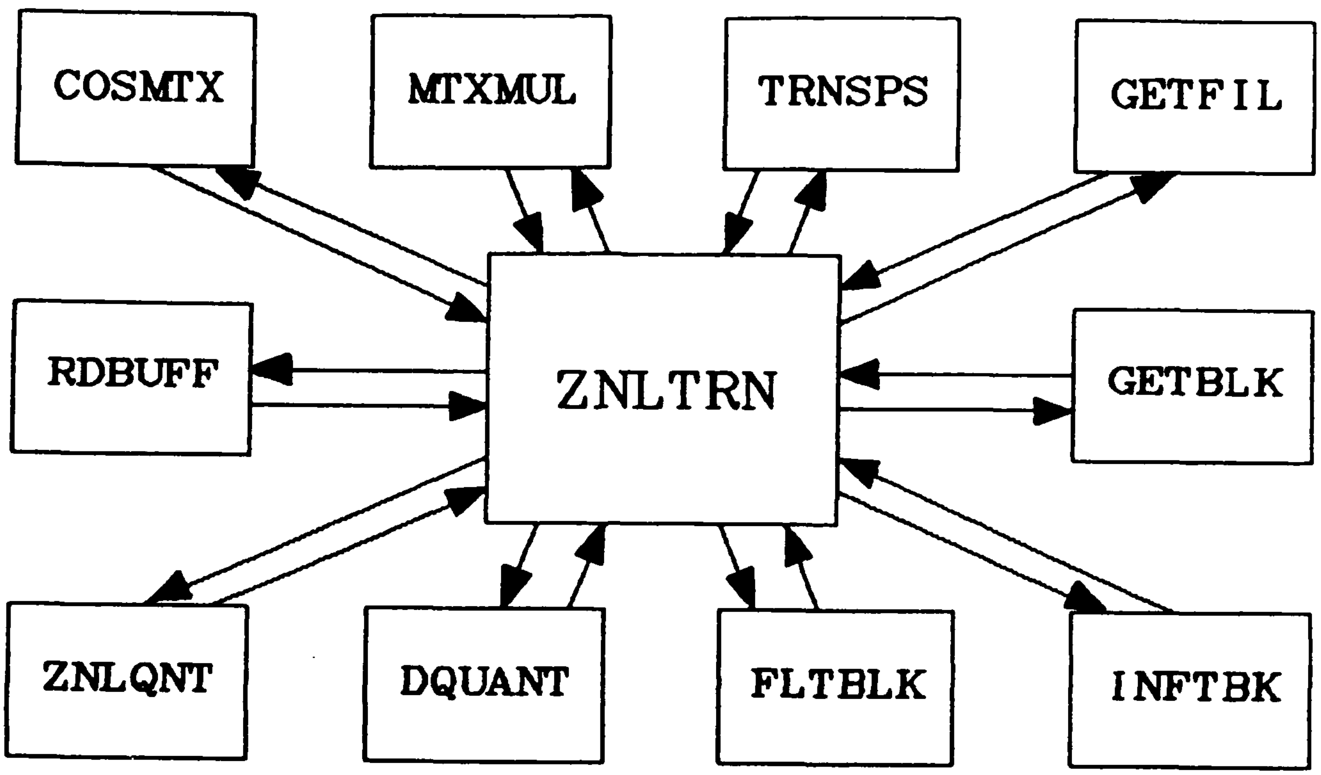

FIGURE A.7 Structure Chart for Module: ZNLTRN

ZNLTRN.FTN - Discrete Cosine Transform Program using the Zonal Quantizing Method

OPEN files and read input parameters

Do 70 times

- Read in the ordering in which the coefficents will be checked in the quantizing routine and their dividing factors

Calculate Cosine Matrix

Calculate Transpose of Cosine Matrix

Define the boundaries of the image file

Initialize the total pixel being processed to zero

Do for the number of vertical sub-blocks

- Get row of horizontal sub-blocks
- Do for the number of horizontal sub-blocks
  - Get sub-block and transform the binary numbers of a sub-block of the image file to real numbers
  - Perform matrix multiplications to transform sub-block matrix
  - Quantize the transformed sub-block using the Zonal Quantization Method
  - Perform an integer filtering routine that puts back in range out of range coefficients that were due to quantization error

ZNLTRN.FTN - Discrete Cosine Transform Program using the Zonal Quantizing Method

Dequantize the transformed sub-block

Perform matrix multiplications to transform the sub-block back to original form. (The sub-block will not be exactly the same due to quantization error)

Perform filtering process that puts back in range the out of range pixels that were due to quantization error

Do for the number of vertical pixels

Do for the number of horizontal pixels

Do for the number of bits per word

Pack bits into word

Increment total pixels being processed

Write a row of horizontal sub-blocks to output file

Calculate and print out compression statistics

E N D

RDBUFF.FTN - Buffer Reading Subroutine

Move down to proper spot reading point of Image file
Do for the number of vertical dimension of sub-blocks   Read horizontal block of pixels
Return
E N D

GETBLK.FTN - Sub-block Retrieving Subroutine

```
Move across to proper spot in buffer
Do for the number of vertical dimension of sub-blocks
    Do for half the horizontal dimension of sub-blocks
        Do two times
            Take half a word which is one pixel from the buffer
Return
E N D
```

ZNLQNT.FTN - Quantizing Subroutine

Initialize Quantizing sub-block

Initialize current coefficients being processed

Do WHILE current coefficient count of the sub-block is less than the user inputted number of coefficients kept and that this is less than 70. (The reason the current coefficient must be less than the seventieth coefficient is that the image distorts keeping more than 70 coefficients.)

- Divide the current coefficient to send it in eight bits
- Increment PP to keep track of coefficients being processed

Since the PP count is always one ahead subtract one to keep precise count

Return

E N D

DQUANT.FTN - Dequantizing Subroutine

Initialize sub-block
Do for the number of pixels processed in the Quantizing Matrix   Multiply current pixel by what it was divided by in the Quantizing Subroutine
Return
E N D

INFTBK.FTN - Filtering Subroutine

Do for vertical dimension of sub-block

  Do for horizontal dimension of sub-block

    Is this the DC coefficient and is it greater than "255"?

    YES

    Make the DC coefficient equal to "255"

    Is this an AC coefficient and is it greater than "127"?

    YES

    Make the current AC coefficient equal to "127"

    Is this an AC coefficient and is it less than "-127"?

    YES

    Make the current AC coefficient equal to "-127"

Return

E N D

Program Documentation for module: ZNLTRN

PROGRAM: ZNLTRN

DESCRIPTION: This program uses a conditional zonal coding technique which employs the Discrete Cosine Transform (DCT). The program will divide an image into sub-block matrices, then transform each sub-block. The transformation process packs the energy into the upper left portion of the matrix. The program interactively inquires for, then accepts an input parameter used in runs quantizing a different amount of coefficients from each sub-block. The quantization process used is the conditional zonal quantizing method. The order in which the coefficients are checked is based upon the highest variances from a cross section of images. After the quantization process the program dequantizes, transforms the sub-blocks back and writes the reconstructed image to an output file. A summary of each run is printed including; names, ending values and compression statistics.

RUNSTRING: ZNLTRN,<INPUT NAME>,<OUTPUT NAME>,<STAT FILE>

INPUT NAME	Input image file name
OUTPUT NAME	Output reconstructed image file name
STAT FILE	Statistics file ordered to check sub-block matrices

INPUT PARAMETER: Ending value

MODULES CALLED:

RDBUFF	Subroutine to read a horizontal line of the input image into the FTN77 buffer.
COSMTX	Subroutine to put in the cosine matrix.
TRNSPS	Subroutine to put in the transpose of the cosine matrix.

Program Documentation for module: ZNLTRN

MTXMUL	Subroutine to do matrix multiplications of real numbers.
GETBLK	Subroutine to retrieve a block of data from the FTN77 buffer.
ZNLQNT	Subroutine to quantize blocks of data.
DQUANT	Subroutine to dequantize blocks of data.
FLTBLK	Subroutine to filter out, out of range real pixels.
INFTBK	Subroutine to filter out, out of range integer coefficients.
GETFIL	Subroutine to open input image an file for processing.

NAMED COMMON DESCRIPTIONS:

Block Name: GFMBLK
Module Common to: RDBUFF

Descriptions:

IMGFIL	Input image file name
EXISTS	File exists flag
ISTAT	File status variable
RECLEN	Record length in bytes
NUMREC	Number of records in input file
RECRDS	Number of records in primary file
FTN77	Fortran read buffer
TEMBUF	Temporary read buffer
ACCTYP	File access flag

Program Documentation for module: ZNLTRN

Block Name:	GTBLK
Module Common to:	RDBUFF,GETBLK

Descriptions:

OUTBUF	Output buffer

Subroutine Documentation for module: RDBUFF

SUBROUTINE: RDBUFF

MODULES CALLED FROM: ZNLTRN, THRTRN

PURPOSE: This subroutine reads a horizontal sub-block of data from the image file into the FTN77 buffer.

MODULES CALLED:

LGBUF	Subroutine to make the buffer size larger.

CALLING FORMAT: CALL RDBUFF(YDIM,YVAL,INLU)

ARGUMENT DESCRIPTIONS:

YDIM	Y dimension of the buffer in words
YVAL	Y coordinate of file for reading
INLU	LU for the input image file

NAMED COMMON DESCRIPTIONS:

Block Name: GFMBLK
Module Common to: ZNLTRN, THRTRN

Descriptions:

IMGFIL	Input image file name
EXISTS	File exists flag
ISTAT	File status variable
RECLEN	Record length in bytes
NUMREC	Number of records in input file

Subroutine Documentation for module: RDBUFF

RECRDS	Number of records in primary file
FTN77	Fortran read buffer
TEMBUF	Temporary read buffer
ACCTYP	File access flag

Block Name: GTBLK
Module Common to: GETBLK, ZNLTRN, THRTRN

Description:

OUTBUF	Output buffer

Subroutine Documentation for module: GETBLK

SUBROUTINE: GETBLK

MODULES CALLED FROM: ZNLTRN, THRTRN

PURPOSE: This subroutine retrieves a block of data from the block buffer and places it in the transform data buffer for transformation.

CALLING FORMAT: CALL GETBLK(XVAL,YVAL,XSIZ,YSIZ,BLKNAM)

ARGUMENT DESCRIPTIONS:

XVAL	Upper left X file coordinate
YVAL	Upper left Y file coordinate
XSIZ	X Block dimension
YSIZ	Y Block dimensio..
BLKNAM	Memory to hold a block of data to be retrieved

NAMED COMMON DESCRIPTIONS:

Block Name: GTBLK
Module Common to: GETBLK, ZNLTRN, THRTRN

Description:

OUTBUF	Output buffer

Subroutine Documentation for module: ZNLQNT

SUBROUTINE: ZNLQNT

MODULE CALLED FROM: ZNLTRN

PURPOSE: This subroutine will quantize a sub-block of pixels. The quantization process takes a 32 bit real coefficient from the buffer and transforms it into an 8 bit integer coefficient. The subroutine quantizes from the upper left portion of the sub-block keeping an inputted number of pixels from each sub-block.

CALLING FORMAT:
```
CALL ZNLQNT(QNTMTX,BLKBUF,F,S,D,LAST,PP,
            XDIM,YDIM)
```

ARGUMENT DESCRIPTIONS:

QNTMTX	Output quantized matrix
BLKBUF	Input matrix to be quantized
F,S	The ordering in which a sub-block of data will be checked (e.g. IF BLKBUF(F,S) .LE. LAST)
D	An array to hold division numbers that convert the 32 bit real numbers into 8 bit integers
LAST	The inputted number of coefficients kept in each sub-block
PP	Keeps count of the pixels being processed in each call to ZNLQNT and sends it to the dequantizing subroutine
XDIM	X Dimension of inputted matrix
YDIM	Y Dimension of inputted matrix

Subroutine Documentation for module: DQUANT

SUBROUTINE: DQUANT

MODULES CALLED FROM: ZNLTRN, THRTRN

PURPOSE: This subroutine will dequantize a transformed quantized sub-block of pixels. The dequantization process takes an 8 bit integer pixel from the quantizing routine and dequantizes it into a 32 bit real pixel. The subroutine dequantizes in the same way the quantization process was done either adaptively or conditionally and using the ordering that was used in the quantizing routine.

CALLING FORMAT: CALL DQUANT(QNTMTX,BLKBUF,F,S,D,PP,XDIM,YDIM)

ARGUMENT DESCRIPTIONS:

QNTMTX	Output quantized matrix
BLKBUF	Input matrix to be dequantized
F,S	The ordering in which a sub-block of data will be checked
D	An array to hold division numbers that convert the 8 bit integer numbers back into 32 bit real numbers
PP	Pixels to be processed that was determined in the quantizing routine
XDIM	X Dimension of inputted matrix
YDIM	Y Dimension of inputted matrix

Subroutine Documentation for module: INFTBK

SUBROUTINE: INFTBK

MODULE CALLED FROM: ZNLTRN, THRTRN

PURPOSE: This subroutine will filter out reconstructed integer coefficients that are out of range. (e.g. If the coefficient is the DC coefficient it can be no larger than 255. Any other coefficient can be no larger than 127 or smaller than -127.)

CALLING FORMAT: CALL INFTBK(MATRIX,DIM)

ARGUMENT DESCRIPTIONS:

MATRIX	Matrix to be filtered
DIM	Dimension of matrix to be filtered

A.2.2.2 CODING PROGRAM: THRTRN

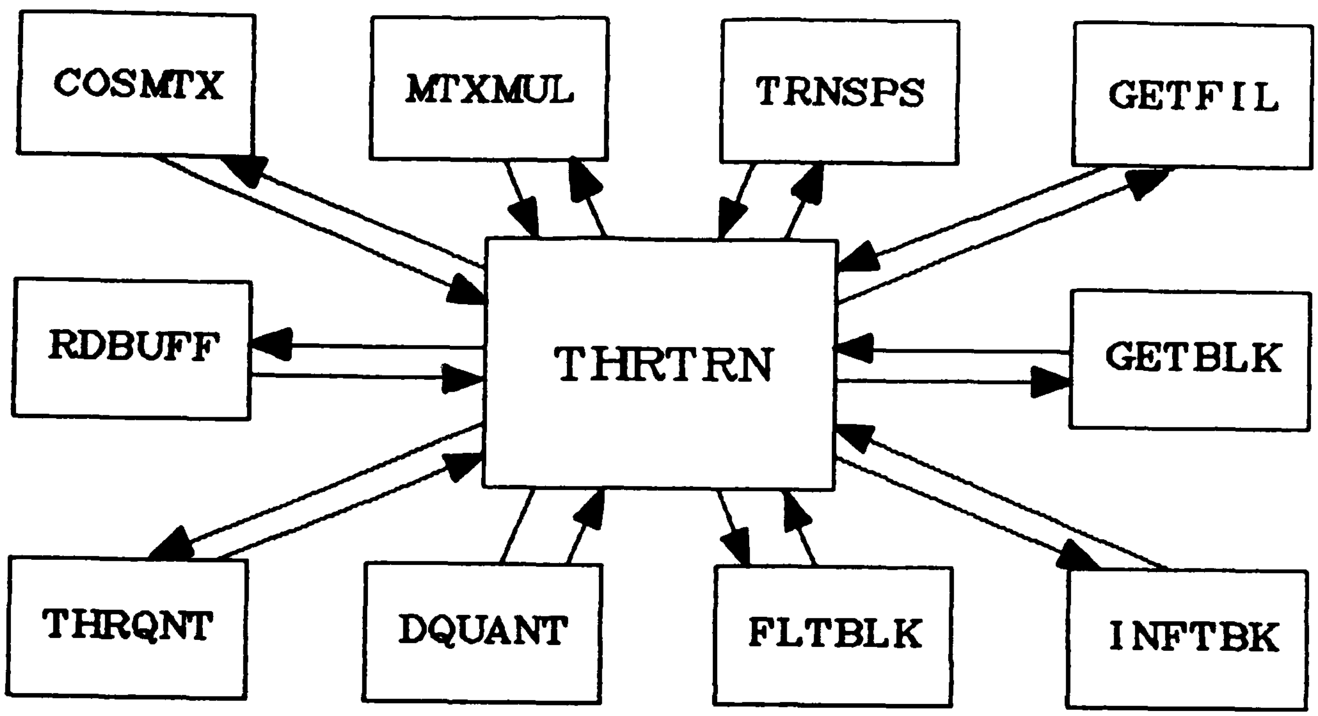

FIGURE A.8 Structure Chart for Module: THRTRN

THRTRN.FTN - Discrete Cosine Transform Program
using the Adaptive Zonal Quantizing Method

OPEN files and read input parameters

Do 70 times

- Read in the ordering in which the coefficents will be checked in the quantizing routine and their dividing factors

Calculate Cosine Matrix

Calculate Transpose of Cosine Matrix

Define the boundaries of the image file

Initialize the total pixel being processed to zero

Do for the number of vertical sub-blocks

- Get row of horizontal sub-blocks
- Do for the number of horizontal sub-blocks
  - Get sub-block and transform the binary numbers of a sub-block of the image file to real numbers
  - Perform matrix multiplications to transform sub-block matrix
  - Quantize the transformed sub-block using the Adaptive Zonal Quantizing Method
  - Perform an integer filtering routine that puts back in range out of range coefficients that were due to quantization error

THRTRN.FTN - Discrete Cosine Transform Program
using the Adaptive Zonal Quantizing Method

- Dequantize the transformed sub-block
- Perform matrix multiplications to transform the sub-block back to original form. (The sub-block will not be exactly the same due to quantization error)
- Perform filtering process that puts back in range the out of range pixels that were due to quantization error
- Do for the number of vertical pixels
  - Do for the number of horizontal pixels
    - Do for the number of bits per word
      - Pack bits into word
- Increment total pixels being processed

Write a row of horizontal sub-blocks to output file

Calculate and print out compression statistics

E N D

THRQNT.FTN - Quantizing Subroutine

Initialize Quantizing sub-block

Initialize current coefficients being processed and FLAG to true

Do WHILE the flag condition is true

> Do WHILE current coefficient of the sub-block is greater than or equal to the cutoff number and that the coefficient is less than 70. (The reason the current coefficient must be less than the seventieth pixel is that the image distorts keeping more than 70 coefficients.)
>
> > Divide the current coefficient to send it in eight bits
> >
> > Increment PP to keep track of coefficients being processed
>
> Are the next two coefficients greater than or equal to 50 times the cutoff number and is the last coefficient less than 70 ?
>
> | YES | NO |
> |---|---|
> | Divide the current coefficient to send it in eight bits | Set flag to false to get out of while loop |
> | Increment PP to keep track of coefficients being processed | |

Since the PP count is always one ahead subtract one to keep precise count

Return

E N D

Program Documentation for module: THRTRN

PROGRAM: THRTRN

DESCRIPTION: This program uses an adaptive zonal coding method which employs the Discrete Cosine Transform (DCT). The program will divide an image into sub-block matrices, then transform each sub-block. The transformation process packs the energy into the upper left portion of the matrix. The program interactively inquires for, then accepts an input parameter used in runs having different quantization cutoff points. The quantization process quantizes coefficients greater than these cutoff points. The order in which the coefficients are checked is based upon the highest variances from a cross section of images. After the quantization process the program dequantizes, transforms the sub-blocks back and writes the reconstructed image to an output file. A summary of each run is printed including; names, cutoff points and compression statistics.

RUNSTRING: THRTRN,<INPUT NAME>,<OUTPUT NAME>,<STAT FILE>

- INPUT NAME — Input image file name
- OUTPUT NAME — Output reconstructed image file name
- STAT FILE — Statistics file ordered to check sub-block matrices

INPUT PARAMETER: Cutoff point

MODULES CALLED:

- RDBUFF — Subroutine to read a horizontal line of the input image into the FTN77 buffer.
- COSMTX — Subroutine to put in the cosine matrix.
- TRNSPS — Subroutine to put in the transpose of the cosine matrix.
- MTXMUL — Subroutine to do matrix multiplications of real numbers.

GETBLK	Subroutine to retrieve a block of data from the FTN77 buffer.
THRQNT	Subroutine to quantize blocks of data.
DQUANT	Subroutine to dequantize blocks of data.
FLTBLK	Subroutine to filter out, out of range real pixels.
INFTBK	Subroutine to filter out, out of range integer coefficients.
GETFIL	Subroutine to open input image an file for processing.

NAMED COMMON DESCRIPTIONS:

Block Name: GFMBLK
Module Common to: RDBUFF

Descriptions:

IMGFIL	Input image file name
EXISTS	File exists flag
ISTAT	File status variable
RECLEN	Record length in bytes
NUMREC	Number of records in input file
RECRDS	Number of records in primary file
FTN77	Fortran read buffer
TEMBUF	Temporary read buffer
ACCTYP	File access flag

Block Name: GTBLK
Module Common to: RDBUFF, GETBLK

Descriptions:

OUTBUF	Output buffer

Subroutine Documentation for module: THRQNT

SUBROUTINE: THRQNT

MODULE CALLED FROM: THRTRN

PURPOSE: This subroutine will quantize a sub-block of pixels. The quantization process takes a 32 bit real coefficient from the buffer and transforms it into an 8 bit integer coefficient. The subroutine quantizes from the upper left portion of the sub-block quantizing coefficients greater than a cutoff point. The quantization process ends when a coefficient is less than the cutoff point as long as it was not an extrinsic coefficient. An extrinsic coefficient is a coefficient that is less than the cutoff point, but the next two coefficients after the extrinsic coefficient are both greater than 50 times the cutoff point. If an extrinsic coefficient was encountered then the quantization process would continue until it fell out of the quantizing routine normally, meeting a coefficient less than the cutoff point that was not an extrinsic coefficient.

CALLING FORMAT:

```
CALL THRQNT(QNTMTX,BLKBUF,F,S,D,CUTOFF,PP,
            XDIM,YDIM)
```

ARGUMENT DESCRIPTIONS:

QNTMTX	Output quantized matrix
BLKBUF	Input matrix to be quantized
F,S	The ordering in which a sub-block of data will be checked (e.g. IF BLKBUF(F,S) .LT. CUTOFF)
D	An array to hold division numbers that convert the 32 bit real numbers into 8 bit integers
CUTOFF	The inputted cutoff point

Subroutine Documentation for module: THRQNT

PP	Keeps count of the pixels being processed in each call to THRQNT and sends it to the dequantizing subroutine
XDIM	X Dimension of inputted matrix
YDIM	Y Dimension of inputted matrix